Inserto a cura di Luca Zaninello

RISERVATO PERSONAL

CBM	**PET Flash** Rendiamo lampeggiante lo schermo dei Commodore.	2
	Logos: un programma creativo Per HP 75: disegni e fantasia.	5
ZX Spectrum	**Poke-Man** Dal quasi omonimo gioco dei bar un divertente programma in BASIC!	9
	PL/Bit: il compilatore 1° La prima parte di un compilatore realizzato dal nostro esperto.	28
TI-99/4A	**Calcolo di strutture** Un programma ingegneristico che risolve problemi di medie dimensioni.	42
COMMODORE 64	**Analisi numerica** Soluzione di equazioni, sistemi, integrali, interpolazioni.	53

SHARP: Regata; pag. 12
CASIO: La titolazione col computer; pag. 26
ZX SPECTRUM: Dizionario; pag. 38
VIC 20: Una partita a golf; pag. 49

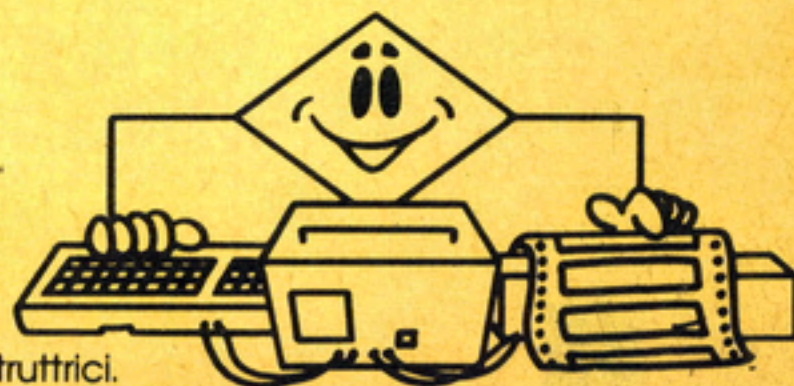
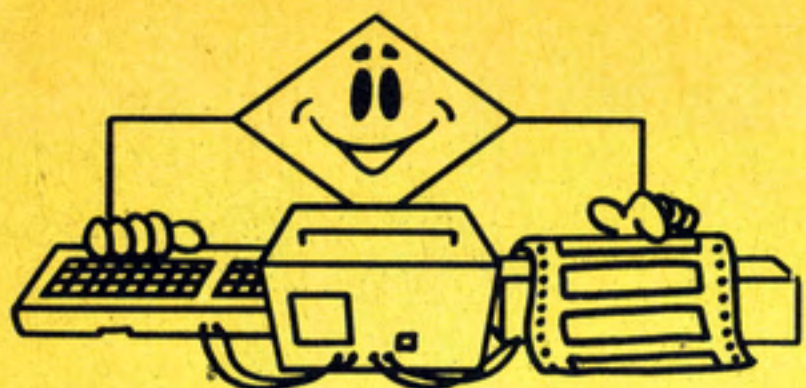
Spedizione in Abb. Postale Gruppo III/70
Questo fascicolo è annesso alla rivista Bit n. 53 Settembre 1984.

PET Flash

Figura 1 - Programmino BASIC che fa lampeggiare una scritta sullo schermo. A qualcuno può bastare!

```
100 PRINT"LAMPEGGIAMENTO STRINGA A$"
110 PRINT"SULLO SCHERMO":PRINT
120 FORK=1TO40:CS$=CS$+CHR$(157):NEXT
130 REM CS$=40 CURSORI A SINISTRA
140 D=200:REM VELOCITA' LAMPEGGIO
150 INPUT"STRINGA";A$:L=LEN(A$)
160 PRINTCHR$(147):PRINT:PRINT:PRINT
170 PRINTTAB(10);:FORN=1TO10
180 PRINTCHR$(18);A$;:FORK=1TOD:NEXT
190 PRINTLEFT$(CS$,L);CHR$(146);A$;
200 FORK=1TOD:NEXT
210 PRINTLEFT$(CS$,L);:NEXTN:PRINT
```

Figura 2 - Caricatore in BASIC del programma in linguaggio macchina. Contiene un esempio di utilizzo.

```
100 PRINT"]***************************"
110 PRINT"█*** F L A S H   C B M ***"
120 PRINT"█***************************"
130 PRINT"INIZIALIZZAZIONE :SYS 30976"
140 PRINT"(DIS)ABILITAZIONE:SYS 30996"
150 PRINT"INIZIO TABELLA   :    31107"
160 INPUT"█E' LA PRIMA ESECUZIONE";R$
170 IFLEFT$(R$,1)="S"THEN200
180 PRINT"E' VIETATO RIESEGUIRE"
190 PRINT"RICARICARE O DISABILITARE":STOP
200 IN=30976:TA=31107
210 FORK=0TO130:READA:POKEIN+K,A
220 S=S+A:NEXT:IFS<>16531THENSTOP
230 FORK=0TO124:POKETA+K,0:NEXT
240 PRINT"CARICATO PROGRAMMA LM."
250 SYS30976:PRINT"INIZIALIZZATO."
260 REM ********************************
270 REM ESEMPIO (DIS)ABILIT. LAMPEGGIO
280 PRINT:PRINT"*** COMANDI ***"
290 PRINT"I:INSERZIONE D:DISABILITAZIONE"
300 PRINT"LAMPEGGIO CELLA SCHERMO"
310 PRINT"DI COORDINATE I,J"
320 PRINT"( DA 0,0 A 24,39 )
330 INPUT"[I/D],RIGA,COLONNA";C$,I,J
340 IFC$<>"I"ANDC$<>"D"THEN330
350 IF(I<0ORI>24)OR(J<0ORJ>39)THEN330
360 TA=31107:T=TA+I*5+INT(J/8)
370 X=2↑(J-INT(J/8)*8)
380 IFC$="I"THENPOKET,PEEK(T)ORX
390 IFC$="D"THENPOKET,PEEK(T)ANDNOTX
400 PRINT"]";:GOTO330
410 REM PROGRAMMA L.M. ***************
420 DATA169,20,141,129,121,141,130,121
430 DATA169,121,133,49,133,53,169,0
440 DATA133,48,133,52,120,174,144,0
450 DATA172,145,0,173,127,121,141,144
460 DATA0,173,128,121,141,145,0,140
470 DATA128,121,142,127,121,88,96,206
480 DATA130,121,208,9,173,129,121,141
490 DATA130,121,32,64,121,108,127,121
500 DATA169,121,133,183,169,131,133,182
510 DATA169,128,133,185,169,0,133,184
520 DATA160,0,162,8,177,182,240,15
530 DATA74,72,144,6,177,184,73,128
540 DATA145,184,104,200,202,208,241,230
550 DATA182,208,2,230,183,165,184,24
560 DATA105,8,133,184,144,2,230,185
570 DATA165,185,201,132,208,210,96
580 DATA47,121:REM INDIRIZZO (L,H) ROUT
590 DATA0,0:REM DELAY,DELCNT
600 REM VOLENDO, ALTRI 125 0
```

di **Edoardo Patrucco**

I calcolatori Commodore non sono dotati di istruzioni per rendere lampeggiante una scrittura sullo schermo. Per ottenere un simile effetto si possono utilizzare sottoprogrammi come quello di figura 1. Essi indubbiamente possono essere utili in molti casi, quando occorra attirare l'attenzione dell'utente di un programma o fargli notare un errore; però fanno perdere alla macchina un tempo che si potrebbe utilizzare meglio e inoltre il lampeggio di scritte in punti lontani sullo schermo è difficoltoso. Con questo programma il lampeggio diventa un'operazione trasparente al programma BASIC e diventa una caratteristica delle locazioni dello schermo; non è però legato ai vari caratteri, ma alla posizione: quindi, ad esempio, i primi dieci caratteri dello schermo lampeggiano sempre, anche dopo uno scorrimento del testo verso l'alto. Questo non è un problema, soprattutto se il programma è usato assieme ad altri che operino senza causare lo scroll della pagina video. Lo stato di lampeggio di ciascuna cella del video si può abilitare o disabilitare e si può eliminare del tutto il lampeggio; l'inversione di tutto lo schermo, a qualsiasi velocità, non è che un caso paticolare. Si opera su una mappa di 125 byte, ciascuno dei quali contiene gli 8 bit che descrivono lo stato normale (bit = 0) o lampeggiante (bit = 1) di 8 celle consecutive di memoria video; per l'8032, che ha 2000 caratteri, occorre una mappa di dimensione doppia ed occorre tenerne conto nell'allocazione del programma in memoria.

Funzionamento del programma

La routine principale, che può essere richiamata separatamente, è la FLASH. Essa legge una per volta le 125 celle della tabella TABLE, ne esamina il contenuto bit a bit, facendolo scorrere a destra, e, se il bit è posto a 1, inverte lo stato normale/inverso del corrispondente carattere sullo schermo. Il test di fine viene fatto sul contenuto del puntatore VIDEO, dopo il suo incremento, quando esso supera il limite della memoria video (che dipende dalla macchina). La routine FLASH è richiamata periodicamente dalla routine ROUT, agganciata al vettore di interruzione della macchina, ogni qualvolta il valore contenuto in DELAY coincide con il corretto numero di cicli di interruzione. Per chi ancora non lo sapesse, i calcolatori Commodore dispongono di un integrato speciale (VIA) che è programmato, tra l'altro, per generare sessanta volte al secondo un segnale, sentito dal processore, che innesca le procedure di gestione della tastiera, aggiornamento della varia-

```
LOC.  OGGETTO   LINEA ..........CODICE SORGENTE...................

                  1  * FLASH SCHERMO COMMODORE
                  2  * EDOARDO PATRUCCO 1983
                  3  * INIZIALIZZAZIONE SYS(30976)
                  4  * SCAMBIO IRQ       SYS(30996)
                  5  * MAPPA TABLE    =      31107
                  6  *
7900              7           *=      $7900
                  8  IRQ      =       $90                 VETTORE INTERRUZIONE
                  9  BASE     =       $B6                 PUNTERA' A TABLE
                 10  VIDEO    =       $B8                 CONTERRA' STVID L;H
                 11  BOSTRI   =       $30                 CIMA STRINGHE
                 12  ENDMEM   =       $34                 FINE MEMORIA
                 13  STVID    =       $8000               INIZIO PAGINA VIDEO
                 14  *
                 15  * INIZIALIZZAZIONE E PROTEZIONE
                 16  *
7900  A9 14      17  INIT     LDA#    20                  VELOCITA' LAMPEGGIO
7902  8D 81 79   18           STA     DELAY               COME PER IL CURSORE
7905  8D 82 79   19           STA     DELCNT
                 20  * AUTO-PROTEZIONE DEL PROGRAMMA
7908  A9 79      21           LDA#    >INIT
790A  85 31      22           STAZ    BOSTRI+1
790C  85 35      23           STAZ    ENDMEM+1
790E  A9 00      24           LDA#    <INIT
7910  85 30      25           STAZ    BOSTRI
7912  85 34      26           STAZ    ENDMEM
                 27  *
                 28  * SCAMBIO VETTORI DI INTERRUZIONE
                 29  * PER ABILITARE/DISABILITARE TUTTO IL LAMPEGGIO
                 30  *
7914  78         31  SWITCH   SEI
7915  AE 90 00   32           LDX     IRQ
7918  AC 91 00   33           LDY     IRQ+1
791B  AD 7F 79   34           LDA     IRQSAV
791E  8D 90 00   35           STA     IRQ
7921  AD 80 79   36           LDA     IRQSAV+1
7924  8D 91 00   37           STA     IRQ+1
7927  8C 80 79   38           STY     IRQSAV+1
792A  8E 7F 79   39           STX     IRQSAV
792D  58         40           CLI
792E  60         41           RTS
                 42  *
                 43  * CONTROLLO NECESSITA' FLASH
                 44  * ESEGUITO AD OGNI CICLO DI INTERRUZIONE
                 45  * L'INDIRIZZO DI ROUT ( CHE ERA IN IRQSAV )
                 46  * E' STATO POSTO IN IRQ;IRQ+1 DA SWITCH
792F  CE 82 79   47  ROUT     DEC     DELCNT
7932  D0 09      48           BNE     FINE
7934  AD 81 79   49           LDA     DELAY
7937  8D 82 79   50           STA     DELCNT
793A  20 40 79   51           JSR     FLASH
793D  6C 7F 79   52  FINE     JMP@    IRQSAV
                 53  *
                 54  * ROUTINE DI LAMPEGGIAMENTO VERA E PROPRIA
                 55  * ( RICHIAMABILE SEPARATAMENTE )
                 56  *
7940  A9 79      57  FLASH    LDA#    >TABLE              INIZIO MAPPA FLASH (H)
7942  85 B7      58           STAZ    BASE+1
7944  A9 83      59           LDA#    <TABLE              INIZIO MAPPA FLASH (L)
7946  85 B6      60           STAZ    BASE
7948  A9 80      61           LDA#    >STVID              INIZIO MEMORIA VIDEO (H)
794A  85 B9      62           STAZ    VIDEO+1
794C  A9 00      63           LDA#    <STVID              INIZIO MEMORIA VIDEO (L)
794E  85 B8      64           STAZ    VIDEO
                 65  *
7950  A0 00      66  LOOP0    LDY#    $00
7952  A2 08      67           LDX#    $08
7954  B1 B6      68           LDA@Y   BASE                LEGGO MAPPA 8 CELLE VIDEO
7956  F0 0F      69           BEQ     NOFLSH              SE =0 PROSEGUO DOPO
7958  4A         70  LOOP     LSRA                        CARRY <-- BIT 0
7959  48         71           PHA                         SALVO ACC. PER DOPO
795A  90 06      72           BCC     DOPOFL
795C  B1 B8      73           LDA@Y   VIDEO               SE CARRY SETTATO
795E  49 80      74           EOR#    $80                 INVERTO NORMALE/REVERSE
7960  91 B8      75           STA@Y   VIDEO
7962  68         76  DOPOFL   PLA
7963  C8         77           INY
7964  CA         78           DEX
7965  D0 F1      79           BNE     LOOP                ESAMINO PROSSIMO BIT
7967  E6 B6      80  NOFLSH   INCZ    BASE                PREPARO INDIRIZZO
```

Figura 3 - Il listato Assembly.

Seguito figura 3.

```
7969  D0 02      81          BNE    NOZER      PROSSIMA CELLA MAPPA
796B  E6 B7      82          INCZ   BASE+1
796D  A5 B8      83  NOZER   LDAZ   VIDEO      PREPARO INDIRIZZO
796F  18         84          CLC               PROSSIME 8 CELLE VIDEO
7970  69 08      85          ADC#   $08
7972  85 B8      86          STAZ   VIDEO
7974  90 02      87          BCC    NOCAR
7976  E6 B9      88          INCZ   VIDEO+1
7978  A5 B9      89  NOCAR   LDAZ   VIDEO+1
797A  C9 84      90          CMP#   $84        =FINE VIDEO ($88 8032)
797C  D0 D2      91          BNE    LOOP0
797E  60         92          RTS
797F  2F         93  IRQSAV  .BY    <ROUT
7980  79         94          .BY    >ROUT
7981  00         95  DELAY   .BY    $00        INIZIALIZZ.CONTATORE
7982  00         96  DELCNT  .BY    $00        CONTATORE
7983  00         97  TABLE   .BY    $00        MAPPA FLASH 1000 CELLE
7A00             98          *=     *+124
                 99  * RISERVATE 125 LOCAZIONI PER 1000 CELLE SCHERMO
                100  * ( PER L'8032 NE OCCORRE IL DOPPIO )
                101  * ( DOVRANNO CONTENERE ALL'INIZIO TUTTI ZERI )
                102  * ( QUI OMESSI PER BREVITA' )
```

SIMBOLO	ESA	VALORE DECIMALE
BASE	00B6	182
BOSTRI	0030	48
DELAY	7981	31105
DELCNT	7982	31106
DOPOFL	7962	31074
ENDMEM	0034	52
FINE	793D	31037
FLASH	7940	31040
INIT	7900	30976
IRQ	0090	144
IRQSAV	797F	31103
LOOP	7958	31064
LOOP0	7950	31056
NOCAR	7978	31096
NOFLSH	7967	31079
NOZER	796D	31085
ROUT	792F	31023
STVID	8000	32768
SWITCH	7914	30996
TABLE	7983	31107
VIDEO	00B8	184

N.B.: sono vietate le uscite anomale da questo stato di scambio di vettori col tasto di RUN/STOP, altrimenti occorre ricaricare in IRQSAV l'indirizzo di ROUT. Logicamente, chi usa il registratore a cassette avrà dei problemi, in quanto il caricamento di programmi o l'uso di file su cassetta richiede che il vettore di interruzione contenga i valori di sistema: è consigliabile in tal caso un'inizializzazione di IRQ con l'indirizzo di ROUT ogni volta.

bile TI ed altre. L'indirizzo di queste procedure "di sistema" è riportato nel cosiddetto vettore di interruzione, ossia nelle due locazioni indicate con IRQ, IRQ + 1 nel listato Assembly (144 nei CBM "grandi", 788 in VIC 20 e Commodore 64).

Per agevolare chi ha già modificato queste locazioni per saltare alla propria, personale, routine di interruzione, la quale per lo più termina con un salto a quelle di sistema (tastiera, ecc.), sono state riservate due celle IRQSAV e IRQSAV + 1 in coda al programma: in esse sarà posto il vecchio contenuto del vettore di interruzione, sia esso un indirizzo di programma utente o l'indirizzo originale, caricato all'accensione della macchina; viceversa, il loro contenuto, inizializzato con l'indirizzo di partenza di ROUT, verrà messo nel vettore di interruzione della macchina IRQ. La routine di inizializzazione INIT; oltre a proteggere il programma, richiama la SWITCH, la quale scambia i contenuti di IRQ e IRQSAV. In questo modo ROUT rimane agganciata alle procedure di interruzione, di sistema o no; da notare che essa termina con un salto indiretto a IRQSAV, che contiene il vecchio vettore di interruzione. Quindi l'utente dovrà eseguire la prima volta INIT; in seguito ogni chiamata a SWITCH inserirà o disabiliterà il lampeggio di tutto lo schermo, senza alterare il regolare svolgersi delle procedure attivate ad ogni ciclo di interrupt (ad esempio accensione delle lampadine sull'albero di Natale!).

Uso del programma

Un esempio di utilizzo è dato all'interno del caricatore BASIC; con esso è possibile abilitare o disabilitare il lampeggio di qualunque cella sullo schermo di un 3032/4032 (per altre macchine occorreranno piccole modifiche sul valore di TABLE, ricavato dall'elenco dei simboli al fondo del listato Assembly, e sul numero di colonne dello schermo).
Si tenga presente che il bit più significativo di una cella della mappa TABLE fa lampeggiare o meno la locazione video più a destra nel gruppo di 8. Spesso però è sufficiente far lampeggiare scritte di 8 ★ n caratteri: ad esempio, assegnata TABLE, la frase POKE TABLE, 255 : POKE TABLE + 1,255 rende lampeggianti i primi 16 caratteri del video.
Si può variare la velocità di lampeggio, modificando il valore di DELAY (inizializzato a 20 ha la velocità del cursore); valori più piccoli accelerano la frequenza. È opportuno prima preparare la mappa e quindi abilitare il lampeggio con SYS (SWITCH), con l'indirizzo SWITCH ricavato dal listato Assembly; per disabilitarlo, verificare che il video sia in stato "NORMAL" prima di eseguire nuovamente la SYS (SWITCH): basta vedere se il contenuto di una locazione lampeggiante è maggiore di 128.

Eventuali modifiche per altre macchine

Il programma può essere agevolmente rilocato, se la zona \$7900-79FF è occupata; vengono utilizzate le locazioni \$B6-\$B9 in pagina zero e 4 byte in coda al programma, oltre alla mappa. La protezione, che può essere alterata, modifica le locazioni di fine memoria BASIC e cima delle stringhe, che sul 64 hanno indirizzi maggiori di 3. Le memorie video sui CBM piccoli iniziano in punti diversi; la mappa ha dimensione pari a numero celle video/8.

Logos: un programma creativo

di **Valerio Anselmo**

Un aspetto forse finora poco noto dell'HP-75 è che esso permette di creare programmi che a loro volta generano altri programmi con la capacità di ripetere all'infinito il processo: programmi che generano programmi che generano programmi ecc. Una specie di realizzazione embrionale delle fantastiche macchine futuristiche che si riproducono autonomamente.

Per riuscire a fare questo basta sfruttare appieno le capacità dell'HP-75 di accumulare in memoria più archivi di vario tipo, di fondere due archivi con MERGE e di saper trasformare un archivio di un certo tipo in uno diverso.

Mettiamo, ad esempio, in macchina il programmino seguente chiamandolo "A". Questo programma fa alcune cose abbastanza semplici: ci fa sapere il proprio nome, ci dice quale programma ha cancellato dalla memoria (il programma "padre"), suona tre note trasmessegli dal "padre" come bagaglio genetico suo proprio, e infine, usando MERGE, genera un programma "figlio", avviandolo e di conseguenza eliminandosi per mano del "figlio" (listato 1).

Una volta avviato, "A" genera "B" dandogli un patrimonio "genetico" leggermente diverso dal proprio, la diversità essendo stata per semplicità ridotta ai dati sempre diversi della riga 150 (la musichetta). "B" cancellerà "A" e genererà a sua volta "C". "C" cancellerà "B" generando "D", e così via.

Un ampliamento delle possibilità di generazione da programma di altri programmi si ha poi sfruttando la capacità della macchina di trasformare, come s'è detto, un archivio di un certo tipo in un archivio di tipo diverso. Le istruzioni che operano la trasformazione sono, come noto, TRANSFORM INTO BASIC, per trasformare un archivio di testo o un archivio LIF1 in un archivio BASIC, TRANSFORM INTO TEXT per trasformare un archivio BASIC o LIF1 in un archivio di testo, e infine TRANSFORM INTO LIF1 per trasformare un archivio BASIC o un archivio di testo in un archivio di interscambio, per lo scambio di informazioni tra l'HP-75 e altri computer.

Il programma Logos che viene presentato in questo numero si serve di TRANSFORM per creare automaticamente un programmino che stamperà un disegno o un marchio, quando sia avviato. Quest'ultimo programma "figlio" potrà essere richiamato con CALL da un programma principale e in tal caso potrà servire ad esempio per

Figura 1 - Disegno realizzato con Logos.

stampare il marchio di una ditta come intestazione di una lettera tramite la stampante HP-82905B. Chi fosse in possesso di un'altra stampante dovrà apportare le poche modifiche necessarie ai comandi che servono per far passare la stampante nel modo grafico. Quando si sarà risposto a tutte le domande del programma ci troveremo ad avere in macchina un altro programma prodotto dal primo, già pronto per l'uso e già provato. Un programma ne avrà generato un altro e lo avrà generato tagliato su misura per i compiti che gli sono richiesti.

È chiaro che queste capacità creative del computer unite alle sue possibilità di comunicazione con periferiche, strumenti di misura, servomotori, ecc., aprono vasti orizzonti alla fantasia del programmatore: si pensi solo alla possibilità di ideare programmi che, in seguito all'insorgere o al mutare di certe condizioni, generino una copia di se stessi più adatta ad affrontare i nuovi problemi, copia che verrà avviata dal programma "padre" (o madre?) e che cancellerà dalla memoria il programma vecchio, ormai inutile; programmi che ne creano altri capaci di generarne a loro volta altri ancora con piccole variazioni casuali; programmi "padre" che generano programmi "figlio" controllandone poi il funzionamento e modificandoli all'occorrenza.

Logos parte da un programma per la creazione di matrici di caratteri già presentato su **Bit**. Ne ripete all'incirca l'impostazione per quanto riguarda l'immissione dei "bit" (quadratini anneriti del disegno su carta millimetrata), anche se qua e là si è operato qualche miglioramento. Nuova è invece la parte che scrive il programma "figlio" e lo prova (istruzioni 270-360). Esaminiamolo ora in dettaglio.

Il programma è lungo 1.300 byte non inizializzato. Ciò significa che potrà essere registrato su una sola scheda magnetica. Le istruzioni di apertura sono classiche: dopo il titolo, nella riga 20 ci sono i dimensionamenti, poi fino alla riga 80 le prime immissioni di dati.

A( ), il vettore che dovrà accogliere le somme dei bit per colonna per ogni striscia di otto punti in

HP

Listato 1 - Un esempio di programma che si riproduce.

```
10 DELAY 2 @ INTEGER X,Y,Z
20 READ B$ @ C$=CHR$(NUM(B$)+1) @ A$=CHR$(NUM(B$)-1)
30 IF A$='@' THEN A$='Z'
40 IF C$="[" THEN C$="A"
50 ON ERROR GOTO 70
60 PURGE A$
70 OFF ERROR
80 ASSIGN # 1 TO C$ @ PRINT # 1,170 ; C$
90 DISP 'SONO IL PROGRAMMA "'&B$&'"' @ DISP 'HO CANCELLATO "'&A$&'"'
100 DISP 'GENERO "'&C$&'"'
110 READ X,Y,Z @ BEEP X,.2 @ BEEP Y,.2 @ BEEP Z,.2
120 RANDOMIZE @ X=500+RND*1000 @ Y=500+RND*1000 @ Z=500+RND*1000
130 PRINT # 1,180 ; X,Y,Z
140 EDIT C$
150 DEF KEY CHR$(255),"MERGE '"&B$&"',10,160 @ RUN '"&C$&"'"
160 PUT CHR$(255)
170 DATA 'A'
180 DATA 500,750,1000
```

verticale, è dimensionato per 94 numeri, tanti quante sono le colonne permesse. E lo stesso dimensionamento subisce la variabile alfanumerica Z$ che rappresenterà visivamente con tanti trattini la serie di quadratini vuoti di una fila del disegno che dovremo far riprodurre dalla stampante.

La prima scelta è tra uso del solo visore dell'HP-75 o uso di uno schermo esterno (un monitor o il televisore di casa) collegato tramite l'interfaccia HP-82163B. Se stiamo usando l'interfaccia video, battere TV dopo i due punti; se si usa solo il visore, battere un asterisco (*) sui due punti e premere RTN.

La denominazione del disegno, che costituirà il nome di un archivio, viene dimensionata ad otto caratteri per non causare un errore (riga 40). La riga 50 apre un archivio di testo con il nome del disegno, archivio che, una volta trasformato in BASIC, costituirà il nucleo del programma "figlio". Con il nome "ZZZZ" apre poi un archivio BASIC per l'immissione da programma dei dati che serviranno per la stampa del disegno. La scelta del nome "ZZZZ" è stata dettata dalla necessità di non interferire con altri programmi eventualmente presenti in memoria. Un archivio col nome "ZZZZ" è statisticamente poco probabile; se però ve ne fosse uno in memoria, occorrerà cambiargli nome (con RENAME) prima dell'avvio di Logos. L'adozione di un archivio BASIC separato per i dati è stata dettata dal fatto che un archivio di testo nella trasformazione in BASIC avrebbe potuto causare errori di sintassi qualora fra i dati fossero state presenti le virgolette. I due programmi, quello col nome del disegno e quello chiamato "ZZZZ", verranno poi comunque fusi in uno alla fine e "ZZZZ" verrà cancellato dalla memoria.

La riga 60 memorizza il numero delle colonne del disegno, cioè il numero di colonne verticali larghe un punto che costituiscono il disegno. Nel caso in cui si usi l'interfaccia video e si voglia avere sullo schermo una rappresentazione visiva immediata del disegno man mano che si forma, sarà bene limitare a 31 il numero massimo delle colonne. La riga 70, poi, richiede il numero di strisce orizzontali (costituite da 8 punti in verticale = 1 byte) che formano il disegno, e la riga 80 chiede di quanto dovrà essere spostato a destra rispetto al margine sinistro della carta il disegno quando verrà stampato.

Si sono esaurite così le operazioni preliminari.

Se usiamo l'interfaccia video e il disegno da riprodurre è semplice, possiamo anche tentare di disegnare direttamente sullo schermo con i tasti "–" e "+". Se invece si usa solo il visore dell'HP-75, o ciò che dev'essere riprodotto è piuttosto complesso, occorrerà preparare in precedenza il disegno su un foglio di carta quadrettata o millimetrata, annerendo un quadratino alla volta quando necessario. Avremo poi suddiviso la superficie interessata in strisce orizzontali alte otto punti, separandole l'una dall'altra con una riga, e avremo contato il numero di colonne verticali, che non dovranno essere più di 94.

Il programma presenterà, come s'è detto, una riga di quadratini vuoti per volta, in attesa dell'immissione. Un quadratino vuoto sarà rappresentato da un meno (–), mentre un quadratino pieno (o annerito nel disegno) dovrà essere rappresentato da un più (+, sottolineato sul visore, in negativo sullo schermo). Per facilitare l'immissione della riga si è assegnato al tasto TAB le stesse funzioni di RTN. TAB si trova vicino ai tasti "–" e "+", e potrà essere premuto più facilmente di RTN. La riga 90 provvede alla ridefinizione del suddetto tasto. Per ottenere il simbolo indicato (la lettera greca "tau" sottolineata) premere SHIFT I/R e poi TAB.

L'uso della forma abbreviata dei comandi (come in questo caso) è stata imposta dalla necessità di mantenere il programma entro i limiti dei 1.300 byte, per poterlo poi registrare su una sola schedina magnetica.

Nella riga 100 si porta la larghezza della riga di stampa all'infinito, per evitare che un disegno durante la stampa possa poi venire spezzato in due nel senso della larghezza. Si riporta poi la stampante nelle condizioni iniziali e si stabilisce un'interlinea adatta a far stampare senza soluzione di continuità le strisce di 8 punti l'una sotto l'altra (9 righe per pollice). Il numerino "27" sottolineato nella riga 100 indica il carattere di fuga ("escape") ottenibile con CTL BACK.

La riga 110 serve a creare la rappresentazione visiva della fila di quadratini vuoti che verrà presentata come suggerimento per l'immissione.

Nella riga successiva (120) si azzera la variabile numerica L, rappresentante la lunghezza della scritta più lunga che viene aggiunta al disegno, e si avvia la prima parte di un'iterazione FOR ... NEXT che conta il numero di strisce orizzontali. La variabile numerica B darà di volta in volta il numero di riga al quale andrà registrata la stringa dei dati in

```
10 ! LOGOS
20 DIM A(94),A$[100],B$[135],Z$[94]
30 DELAY 0 @ INPUT "display is ",":";S$
40 D$="" @ INPUT "nome disegno: ";D$[1,8]
50 ASSIGN # 1 TO D$,TEXT @ ASSIGN # 2 TO "ZZZZ"
60 INPUT "num. colonne (max 94): ";C
70 INPUT "num. strisce orizz.: ";S
80 INPUT "spostam. dal margine: ";M
90 DEF KEY 'I̲',CHR$(13)
100 PWIDTH INF @ PRINT "␛E␛&19D"
110 Z$="" @ FOR A=1 TO C @ Z$=Z$&"-" @ NEXT A
120 L=0 @ FOR A=1 TO S @ B=70+A*10
130 DISPLAY IS S$ @ MARGIN C @ FOR I=1 TO C @ A(I)=0 @ NEXT I
140 DEF KEY '+','(̲±'; @ FOR I=1 TO 8
150 IF S$='*' THEN DISP "riga";I;"della striscia";A @ WAIT 1
160 BEEP 500 @ INPUT "",Z$;A$
170 FOR J=1 TO C @ IF A$[J,J]='±' THEN A(J)=A(J)+2^(8-I)
180 NEXT J @ NEXT I
190 B$="␛*b"&STR$(C)&"G"
200 FOR I=1 TO C @ B$=B$&CHR$(A(I)) @ NEXT I @ B$=B$&" "
210 DISPLAY IS * @ MARGIN 33 @ DISP "scritta (";A;"): ";
220 DEF KEY '+','+'; @ DEF KEY '├̲','""~[̲'; @ PUT '├̲'
230 BEEP 750 @ INPUT ""; F$ @ B$=B$&F$ @ IF LEN(F$)>L THEN L=LEN(F$)
240 PRINT B$;"           striscia";A @ PRINT @ PRINT @ PRINT @ PRINT
250 INPUT "va bene? ","s"; C$ @ IF C$#"s" THEN 130
260 PRINT # 2,B ; B$ @ NEXT A @ DISPLAY IS S$
270 PRINT # 1,10 ; "DIM A$["&STR$(5+C+LEN(STR$(C))+L)&"]"
280 PRINT # 1,20 ; "PWIDTH INF @ PRINT '␛E␛&19D'"
290 IF M=0 THEN F$="PRINT A$" ELSE F$="PRINT TAB("&STR$(M+1)&");A$"
300 IF S=1 THEN PRINT # 1,50 ; "READ A$ @ "&F$ @ GOTO 340
310 PRINT # 1,40 ; "FOR I=1 TO "&STR$(S)
320 PRINT # 1,50 ; "READ A$ @ "&F$
330 PRINT # 1,60 ; "NEXT I"
340 PRINT # 1,70 ; "PRINT '␛E'"
350 TRANSFORM D$ INTO BASIC @ EDIT D$ @ MERGE "ZZZZ" @ RENUMBER @ PURGE "ZZZZ"
360 CALL D$ @ DEF KEY 'I̲','I̲'; @ DEF KEY '├̲','├̲'; @ DISP
```

Listato 2 - Il programma Logos.

"ZZZZ". Tale numero parte da 80 in quanto dovrà essere superiore al numero dell'ultima istruzione del programma col nome del disegno quando "ZZZZ" sarà fuso con questo.
Per poterci accorgere ad orecchio che si è battuto l'ultimo carattere della riga si fissa il margine alla lunghezza della stringa suggerita per l'immissione (riga 130). In tal modo, anche senza alzar la testa dal foglio, avremo un controllo dell'immissione quanto mai utile. Nella stessa riga 130 viene azzerata la porzione di vettore che interessa. L'istruzione DISPLAY IS S$ riattiva il video esterno quando si devono immettere le strisce successive.
Si definisce ora il tasto "+" come "(̲+" (ottenibile con SHIFT I/R SHIFT I/R SHIFT I/R SHIFT CLR) e si inizia la richiesta di immissione dei "bit" del disegno, effettuabile tramite i tasti "−" (meno) e "+" (più), come s'è detto prima. Per poter disporre di tutta la lunghezza della riga di visualizzazione quando si opera l'immissione, si visualizza per un secondo (se l'uscita è sul solo visore) il numero di riga e quello della striscia (riga 150) e quindi si presenta separatamente la fila dei "−", che potranno arrivare così fino a 94. Prima di far questo, però, si segnala con un segnale acustico basso (500 Hz) la possibilità di iniziare ad immettere i dati, e ciò sempre per aiutare l'operatore che se ne sta probabilmente chinato sul foglio di carta millimetrata con una grossa lente in mano cercando di non perdere il segno di dove è arrivato. Quando sente il "beep" basso, sa che può partire con l'immissione dei più (+) e dei meno (−), che diventerà automatica dopo un po'.

Una volta completata ed immessa la "stringa" di "−" e "±" rappresentante la fila di quadratini vuoti e pieni, la riga 170 ne calcola il valore come "bit" e lo va ad aggiungere ai valori già presenti nel vettore A().
Immesse tutte e otto le file che costituiscono una striscia, i dati accumulati in A() vengono trasformati in un comando che potrà far passare la stampante nel modo grafico (righe 190 e 200).
Dopo ogni striscia di disegno si possono far stampare delle scritte, che saranno posizionate ad uno spazio dal disegno. La variabile che accetterà la scritta (F$) non è stata dimensionata, per cui potrà accogliere solo fino a 32 caratteri. Se supereremo tale lunghezza, però, non succederà niente di male: il computer ce lo segnalerà e ripresenterà la scritta per la correzione.
Nella riga 210 un'istruzione MARGIN farà sì che venga segnalata acusticamente con un "beep" la battuta del 32° carattere. Viene quindi visualizzato, solo sul visore dell'HP-75 e non sul video, il messaggio di sollecitazione (formato dalla parola "scritta", con il numero d'ordine della striscia tra parentesi), seguito da due virgolette con il cursore di inserimento sulla seconda virgoletta. Quest'artificio permette di immettere, oltre a degli spazi iniziali, anche le virgole, che altrimenti darebbero luogo ad un errore essendo considerate dal computer come separatori fra più "input" successivi. La riga 220 riporta il tasto "+" al suo giusto valore e definisce un "tasto" non accessibile da tastiera (quello corrispondente al carattere numero 255) nel modo seguente. Si dà al suddet-

Listato 3 - Le variazioni per la stampante MX-80.

```
40 D$="" @ INPUT "nome : ";D$[1,8]
100 PWIDTH INF @ PRINT "␛@␛A"&CHR$(8)
190 B$="␛K"&CHR$(C)&"△"
280 PRINT # 1,20 ; "PWIDTH INF @ PRINT '␛@␛A'&CHR$(8)"
340 PRINT # 1,70 ; "PRINT '␛@'"
```

to "tasto" il valore di due virgolette (" ") seguite dai comandi di cambiamento del cursore (carattere numero 136, equivalente alla pressione del tasto I/R) e spostamento del cursore a sinistra (carattere numero 134, equivalente alla pressione del tasto con la freccia a sinistra). I contenuti del "tasto" numero 255 vengono resi attivi dall'istruzione PUT posta immediatamente prima di un INPUT con la prima parte (il "messaggio di sollecitazione") nulla. Tra parentesi, ricordiamo che, per poter scrivere il carattere numero 255, occorre prima definire un tasto (ad esempio con DEF KEY "#", CHR$(168)**&**CHR$(255);) ed usarlo poi per immettere il carattere nel programma.

Un segnale acustico un po' più acuto (750 Hz) avverte l'operatore che è ora di immettere la scritta (riga 230). Dopo l'immissione della scritta (che potrà anche essere nulla), e l'aggiunta di questa alla variabile B$ contenente già i comandi per la stampa della striscia interessata e uno spazio vuoto, viene effettuato un confronto atto a memorizzare nella variabile L la lunghezza della scritta più lunga fra quante vengono immesse. Viene quindi subito effettuata una prova di stampa della suddetta striscia (riga 240), seguita da otto spazi e dall'indicazione del numero d'ordine della striscia stessa, ciò al fine di permettere una agevole identificazione di quanto viene stampato, che sarà stato portato nel frattempo oltre il bordo metallico del fermacarta della stampante mediante cinque PRINT consecutivi.

La domanda che segue, "va bene?" suggerisce già una risposta "s" per "sì" (riga 250). Se il disegno va bene, basterà premere RTN (o TAB). Se la risposta sarà stata diversa da quella suggerita (come quando si immette "n" per "no"), il programma torna a chiedere i dati per quella striscia, ricominciando dalla riga 130. Se la risposta sarà invece stata "s", la stringa B$ (contenente il comando di passaggio nel modo grafico, i caratteri corrispondenti ai vari "byte" per il comando della testina, lo spazio vuoto e l'eventuale scritta aggiunta) viene registrata nell'archivio BASIC "ZZZZ" mediante il "canale" numero 2, e si continua l'iterazione iniziata alla riga 110, chiudendo il ciclo con l'istruzione NEXT. Usciti dall'iterazione si riattiva per prima cosa il video esterno, se collegato.

Quando nell'archivio "ZZZZ" sono stati accumulati tutti i dati necessari, si passa alla seconda parte del programma, quella che "scriverà" nell'archivio di testo il programma "figlio".

Anche questo programma "figlio" sarà di tipo tradizionale. Si inizia con i dimensionamenti: la riga 270 di Logos attraverso il "canale" numero 1 scrive infatti nella riga 10 dell'archivio di testo che porta il nome del disegno il dimensionamento della variabile A$ che servirà a leggere le varie stringhe corrispondenti alle strisce orizzontali successive del disegno. La lunghezza di tali strisce è determinata dal comando di passaggio al modo grafico (i caratteri "escape", "*", "b" e, dopo il numero dei byte, "G") e lo spazio vuoto aggiunto (nella riga 200) dopo i caratteri componenti il disegno (in totale 5 caratteri), il numero di caratteri che corrispondono ai "byte" della striscia (la variabile C), la lunghezza della stessa variabile C considerata come "stringa" (1 o 2 caratteri) e infine la lunghezza della scritta più lunga fra quelle aggiunte dopo le varie strisce componenti il disegno.

La seconda delle righe che vengono scritte nell'archivio di testo e che porterà il numero 20 non è altro che una duplicazione della riga 100 di Logos. La successiva riga 290 di Logos prepara la scrittura della riga 50 nel programma "figlio". Nella stampa della striscia verrà usata l'istruzione TAB solo se lo spostamento dal margine sinistro per la stampa del disegno è diverso da zero. La riga successiva farà scegliere per il programma "figlio" una versione semplificata della stampa grafica (senza iterazione) se non vi è più di una striscia. In caso contrario verrà usata la versione più completa (righe 310-330). Con la riga 340 si scrive nel programma "figlio" un comando che riporta la stampante nelle condizioni iniziali, con spaziatura normale (6 righe per pollice).

Completata la compilazione del nucleo del programma "figlio", per ora ancora nella forma di archivio di testo privo di vita (una semplice successione di caratteri senza alcun vero significato per il computer), nella riga 350 si opera il sortilegio trasformando questo archivio di testo in un programma in BASIC, in cui le sequenze di caratteri sono ora considerate come "parole" con un preciso significato per la macchina, mentre i numeri delle righe rappresentano la successione temporale in cui la sequenza di istruzioni dev'essere eseguita. A questo nucleo di programma si aggiunge con MERGE la porzione "ZZZZ" già precedentemente in BASIC e si rinumera il tutto, ottenendo così un programmino pulito e perfettamente autonomo per la stampa del disegno da noi preparato.

Dopo aver cancellato "ZZZZ", viene poi immediatamente effettuata (con CALL nella riga 360) una prova di stampa, dopodiché vengono ridefiniti i tasti TAB e CHR$(255) e si termina cancellando il visore.

Come si è detto, il programma funziona così com'è con la stampante HP 82905B. Chi fosse in possesso di una stampante di tipo diverso usandola tramite il convertitore HP-IL dovrà apportare alcune modifiche al programma per adattarlo alla propria stampante. Ad esempio, con una stampante Epson MX-80 TYPE III si dovranno modificare le righe seguenti per poter mantenere il programma entro i 1.300 byte e ottenere lo stesso disegno che si otterrebbe su una HP 82905B (listato 3).

E per finire un avvertimento. Attenzione che, se si interrompe Logos durante l'immissione dei "bit" grafici, il tasto "+" resterà definito per rappresentare il carattere numero 171 (± sul visore). Sarà allora poi necessario ridefinirlo a mano, riportandolo al suo valore iniziale con DEF KEY CHR$(43), CHR$(43); o cancellando l'archivio KEYS con PURGE KEYS.

Poke-Man

di **Lorenzo Covini**

Questo gioco trae ispirazione dal quasi omonimo gioco da bar; scritto interamente in BASIC, ha la possibilità di due velocità.
Lo schermo si presenta diviso orizzontalmente in due parti distinte. In alto, dove inizialmente compare il giallo, "Poke-Man" si presenta con i puntini da mangiare e i tre piccoli fantasmi nemici. In basso, dove i fantasmi non possono mai scendere, ci sono altri prelibati puntini da divorare, ma anche pareti che, diversamente da quelle superiori, sono elettrizzate: andarvi a sbattere contro sarà fatale.
Nel caso vengano mangiati tutti i puntini, e lo schermo completato, si sommerà al punteggio un bonus inversamente proporzionale al tempo impiegato e si potrà ricominciare da capo.
Un'ulteriore difficoltà è data dal non mangiare i puntini: se infatti Poke-Man, per qualsiasi motivo, non continua a mangiare, il punteggio decresce inesorabilmente. È per questo che, iniziando, si parte da 50 punti: infatti se si arriva a zero, il gioco è concluso.

Descrizione del programma

7-8 Presentazione.
9-46 Istruzioni - Inizio del gioco: la linea 11 e la prima parte della 13 definisconò la stringa S$, che viene stampata dalla subroutine delle linee 44-46.
47 Inizializzazione della variabile Hi, che conterrà il record.
50-63 Definizioni caratteri grafici.
99 Inizializzazione variabile P (= punti).
100-101 Richiesta e definizione della velocità della partita.
102-302 Costruzione del labirinto.
305-420 Inizializzazione delle principali variabili del programma.
430 Se il Poke-Man si trova nella metà superiore dello schermo, il programma scende alle linee 5430-5450, che leggono i fattori di incremento delle coordinate dei due fantasmi dalle linee di DATA in fondo al listato, li sommano alle coordinate precedenti, cancellano i fantasmi dalla vecchia posizione e li stampano alla nuova. In più permettono l'accesso ad una seconda subroutine (linee 3200-3220), che muove casualmente il terzo fantasma, controllando (linea 3210) che non finisca sui muri o che non scenda nella parte inferiore del labirinto.

```
LISTATO POKE MAN


   1 REM  "POKE - MAN"
   2 REM  un gioco per
    Sinclair ZX Spectrum 16/48K
   3 REM di Covini Lorenzo,
         Via S.Simpliciano,6
           20121 MILANO
   4 REM   Tel.3058517
   5 REM
   6 BRIGHT 1: BORDER 6: PAPER 6
 : INK 4: CLS
   7 PRINT AT 3,0; INK 2;"

  "" POKE - MAN ""

                                "
   8 FOR q=0 TO 30: BEEP .005,30
 : NEXT q: FOR a=11 TO 21: PRINT
AT a,0; BRIGHT 1; INK 2;"  desi
gned by Lorenzo Covini   ";AT a-
1,0; INK 4; BRIGHT 0;"
                        ": PAUSE 7
 : BEEP .002,20: NEXT a
   9 INPUT FLASH 0; INK 2;"
  "; FLASH 1;"ISTRUZIONI?(s-n)";
 FLASH 0;" "; LINE a$
  10 IF a$<>"s" THEN GO TO 15
  11 CLS : LET s$="          POKE
 - MAN
                             ISTR
UZIONI

                    Tu sei l'orma
i celebre pallina    gialla.
                    Il tuo compit
o e' quello  di      mangiare tut
ti i puntini,evi-    tando nel co
ntempo i 3 terri-    bili fantasm
ini.                Questi,pero',
(bonta'loro) non     scendono mai
 nella parte in-     feriore dell
o schermo(quella     delimitata d
a quelle specie  di    strani ast
erischi).            Qui pero' le
 mura sono elet-       trizzate e
 andarci a sbatte-    re ti sara'
 fatale."
  12 GO SUB 44
  13 LET s$="               I fanta
smini lasciano dietro di se   og
ni tanto dei sosia: sta tranquil
lo,non sono pericolosi!!  Alla f
ine di ogni schermo ti     sara'
dato un bonus proporzio-nalmente
 al tempo impiegato   a  completa
rlo.                          E O
RA BUONA FORTUNA!!           "
  14 GO SUB 44: LET s$="

                               TA
STI di CONTROLLO:

 Q=Sinistra
 W= Destra
 L=  Giu'
 P=   Su                  ": GO SU
B 44
  18 FOR a=0 TO 21: PRINT "
```

Listato 1 - Il listato del gioco per lo Spectrum.

ZX SPECTRUM

Seguito listato 1.

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■": PO
KE 23692,255: NEXT a: PRINT AT 2
1,0; INK 2;          " - premi un t
asto per giocare - "
   20 IF INKEY$="" THEN GO TO 20
   30 GO TO 47
   44 FOR a=1 TO LEN s$: PRINT PA
PER 1; INK 7;s$(a TO a);: BEEP ,
01,25: IF s$(a TO a)=" " THEN BE
EP ,03,20
   45 POKE 23692,2: NEXT a
   46 RETURN
   47 LET hi=0
   50 FOR a=0 TO 7: READ q: POKE
USR "a"+a,q: NEXT a
   51 DATA 255,255,219,231,231,21
9,255,255
   52 FOR a=0 TO 7: READ q: POKE
USR "b"+a,q: NEXT a
   53 DATA 60,126,254,248,240,252
,126,60
   54 FOR a=0 TO 7: READ q: POKE
USR "c"+a,q: NEXT a
   55 DATA 60,126,127,31,15,63,12
6,60
   56 FOR a=0 TO 7: READ q: POKE
USR "d"+a,q: NEXT a
   57 DATA 0,98,231,247,255,255,1
26,60
   58 FOR a=0 TO 7: READ q: POKE
USR "e"+a,q: NEXT a
   59 DATA 60,126,255,255,247,231
,98,0
   60 FOR a=0 TO 7: READ q: POKE
USR "f"+a,q: NEXT a
   61 DATA 124,254,214,254,254,19
6,254,170
   62 FOR a=0 TO 7: READ q: POKE
USR "g"+a,q: NEXT a
   63 DATA 28,42,42,62,42,54,62,4
2
   99 LET p=50
  100 INPUT INK 2;"         VELOCIT
A'?(h-l) "; LINE f$: IF f$<>"h"
AND f$<>"l" THEN GO TO 100
  101 LET f=10: IF f$="l" THEN LE
T f=0
  102 BRIGHT 1: PAPER 5: INK 1: B
ORDER 6: CLS
  105 FOR a=0 TO 21: PRINT AT a,0
;"..............................
..": NEXT a
  110 FOR a=0 TO 31: PRINT AT 0,a
; PAPER 4;"■";AT 21,a; PAPER 6;"
▩"
  120 PRINT AT 12,a; PAPER 4;"■";
AT 14,a; PAPER 6;"▩": NEXT a
  130 PRINT AT 12,8;".";AT 12,23;
".";AT 14,[illegible]".";AT 14,30;".";AT
13,15; PAPER 6;"▩"
  140 FOR a=0 TO 12: PRINT AT a,0
; PAPER 4;"■";AT a,31;"■": NEXT
 a
  145 FOR a=13 TO 21: PRINT AT a,
0; PAPER 6;"▩";AT a,31;"▩": NEXT
 a
  150 FOR a=2 TO 10: PRINT AT a,2
; PAPER 4;"■";AT a,29;"■"
  160 PRINT AT 2,a; PAPER 4;"■";A
T 2,a+19;"■";AT 10,a+19;"■";AT a
,19;"■": NEXT a: PRINT AT 9,19;"
."
  170 FOR a=20 TO 29: PRINT AT 16
,a; PAPER 6;"▩";AT 19,a;"▩": NEX
T a
  180 FOR a=16 TO 19: PRINT AT a,
2; PAPER 6;"▩";AT a,8;"▩";AT a,1
8;"▩";AT a,29;"▩": NEXT a
  190 PRINT AT 18,21; PAPER 6;"▩"
;AT 17,23;"▩";AT 18,25;"▩";AT 17
,27;"▩"
  200 FOR a=14 TO 19: PRINT AT 8,
a; PAPER 4;"■";AT 10,a;"■": NEXT
 a
  210 FOR a=4 TO 10: PRINT AT a,4
; PAPER 4;"■";AT a,5;"■";AT a,6;
"■";AT a-2,14;"■";AT 5,a;"■";AT
6,a;"■"
  220 PRINT AT 4,a; PAPER 4;"■"
  230 NEXT a
  240 FOR a=8 TO 10: PRINT AT a,8
; PAPER 4;"■";AT 8,a;"■": NEXT a
  250 FOR a=2 TO 4: PRINT AT a,21
; PAPER 4;"■";AT a+6,21;"■";AT a
+2,12;"■";AT 16,a+2; PAPER 6;"▩"
;AT 17,a+2;"▩";AT 19,a;"▩": NEXT
 a
  260 FOR a=4 TO 8: PRINT AT a,25
; PAPER 4;"■■■■": NEXT a
  270 PRINT AT 10,10; PAPER 4;"■"
;AT 10,12;"■";AT 8,12;"■";AT 2,1
2;"■";AT 19,6; PAPER 6;"▩";AT 3,
15; PAPER 4;"■";AT 2,a;"■";AT 3,
17;"■";AT 5,16;"■";AT 5,18;"■"
  280 PRINT AT 7,16; PAPER 4;"■";
AT 7,17;"■";AT 6,21;"■";AT 4,23;
"■";AT 6,23;"■";AT 8,23;"■"
  290 FOR a=8 TO 12: PRINT AT 16,
a; PAPER 6;"▩";AT 19,a;"▩";AT 16
,a+6;"▩";AT 19,a+6;"▩": NEXT a
  300 PRINT AT 18,10; PAPER 6;"▩▩
▩";AT 18,14;"▩▩▩"
  302 PRINT #1;AT 1,1;"PUNTI";AT
1,23;"HI ";hi
  305 LET i=0
  306 LET i2=3
  307 LET i3=2
  308 LET c=0
  309 LET q=0
  310 LET x=11: LET y=17
  320 LET x1=3: LET y1=28
  330 LET x2=9: LET y2=28
  340 LET x3=7: LET y3=15
  360 LET sx=0: LET sy=0
  370 LET sx1=0: LET sy1=0
  380 LET sx2=0: LET sy2=0
  390 LET p$="■"
  400 LET a$=SCREEN$ (x1,y1)
  410 LET b$=SCREEN$ (x2,y2)
  420 LET c$=SCREEN$ (x3,y3)
  430 IF x<13 THEN GO SUB 5430
  445 PRINT AT x,y;" "
  450 IF INKEY$="" THEN GO TO 500
  460 IF INKEY$="w" THEN LET sy=1
: LET sx=0: LET p$="■"
  470 IF INKEY$="q" THEN LET sy=-
1: LET sx=0: LET p$="■"
  480 IF INKEY$="l" THEN LET sy=0
: LET sx=1: LET p$="■"
  490 IF INKEY$="p" THEN LET sy=0
: LET sx=-1: LET p$="■"
  500 LET x=x+sx: LET y=y+sy
  505 IF ATTR (x,y)<>105 THEN GO
SUB 4100
  535 IF p<1 THEN LET p=0: GO TO
3050
  550 IF SCREEN$ (x,y)="." THEN L
ET p=p+25: LET c=c+1: BEEP .001,
32
  560 PRINT AT x,y; INK 6;p$;
  561 IF x<13 THEN PRINT AT x1,y1
; INK i;"■";AT x2,y2; INK i2;"■"
;AT x3,y3; INK i3;"■"
  570 IF ATTR (x,y)<>110 THEN GO
SUB 5510
  580 IF x>12 THEN FOR a=0 TO 10-
f: NEXT a
  595 IF c=333 THEN GO TO 3100
  600 LET q=q+1: LET p=p-10: PRIN
T #1;AT 1,7;"     ";AT 1,7;p: GO
TO 430
 3000 BEEP 2,10: FOR a=0 TO 18: B
EEP .01,20: NEXT a: PRINT AT 10,
0; PAPER 1; INK 5;"           P
ECCATO!!
                          Ti h
a pappato!!!         ";AT 21,0;"
   Premi per giocare ancora     "
```

450-500 Lettura della tastiera e aggiornamento della posizione di Poke-Man.
505 Il computer controlla che Poke-Man non si sovrapponga ai muri.
535 Si controlla che il punteggio non sia sceso sotto zero.
550 Se Poke-Man sta mangiando un puntino, il punteggio sale, aumenta di un punto il contatore dei puntini mangiati e viene emesso un suono.
560-1 Vengono stampati Poke-Man e i fantasmi, se Poke-Man è nel labirinto superiore.
570 Se un fantasma si è sovrapposto a Poke-Man, il programma va alle linee 5510-5530, che accertano che sia un fantasma e non uno dei sosia che questi si lasciano dietro.
580 Nel caso il fantasma si trovi nel labirinto inferiore, e la velocità scelta sia la minore, un apposito ciclo FOR...NEXT rallenta l'esecuzione del programma.
600 Vengono aggiornate alcune variabili, stampato il punteggio e il programma torna alla linea 430.
3000-3020 Routine, nel caso Poke-Man venga mangiato da un fantasma.
3050-3070 Routine, nel caso il punteggio scenda sotto zero.
3100-3120 Routine per il BONUS STAGE, nel caso lo schermo venga completato.
3150-3195 Routine nel caso il Poke-Man venga fulminato.
3200 Vedi linea 5435.
4100-4140 Controlla se Poke-Man è finito contro un muro normale o elettrizzato, nel qual caso la partita finisce (vedi linea 505).
5430-5450 Vedi linea 570.
9000-9001 Linee di DATA, che contengono le istruzioni per i movimenti dei primi due fantasmi. Possono essere cambiate a piacere, ma devono sempre terminare con 100, 0, 0, 0, per informare il computer di eseguire il RESTORE.

Seguito listato 1.

```
3005 IF p>hi THEN LET hi=p
3010 IF INKEY$="" THEN GO TO 301
0
3020 RESTORE 9000: GO TO 99
3050 BEEP 2,10: FOR a=0 TO 18: B
EEP .005,30: NEXT a: PRINT AT 10
,0; INK 7; PAPER 1;            "
              PECCATO!!
       Sei sceso sotto zero!!
": PRINT #1;AT 1,7; FLASH 1; INK
 0; PAPER 7;"----": IF p>hi THEN
 LET hi=p
3060 IF INKEY$="" THEN GO TO 306
0
3070 RESTORE 9000: CLS : GO TO 9
9
3100 FOR a=0 TO 40: BEEP .005,36
: NEXT a: BEEP .02,20: BEEP .004
,17: BEEP .007,20: BEEP .04,2
3110 PRINT AT 8,13; PAPER 2; INK
 7; FLASH 1;"BONUS";AT 9,13;"STA
GE": FOR w=0 TO (700-q): PRINT A
T 11,13; FLASH 1; INK 2; PAPER 7
;w*25: BEEP .008,28: NEXT w: LET
 p=p+w*20
3120 FOR a=0 TO 300: NEXT a: CLS
 :;RESTORE 9000: LET f=10: GO TO
 102
3150 BEEP 1,30: FOR a=0 TO 5: FO
R b=0 TO 7: PRINT AT x,y; INK b;
p$: NEXT b: NEXT a
3160 PRINT AT 10,0; INVERSE 1;"
        Sei rimasto

     F U L M I N A T O ! !

     ________________________
                               "
3170 IF p>hi THEN LET hi=p
3180 RESTORE 9000: PRINT AT 21,0
; INVERSE 1;"     -premi per rico
minciare-     "
3190 IF INKEY$="" THEN GO TO 319
0
3195 CLS : GO TO 99
3200 PRINT AT x3,y3;c$: LET sx3=
-1+INT (RND*3): LET sy3=-1+INT (
RND*3)
3205 LET x3=x3+sx3: LET y3=y3+sy
3
3210 IF ATTR (x3,y3)=97 OR ATTR
(x3,y3)=110 THEN LET x3=x3-sx3:
LET y3=y3-sy3: RETURN
3215 LET c$=SCREEN$ (x3,y3)
3216 PRINT AT x3,y3;c$
3220 RETURN
4100 IF ATTR (x,y)=97 THEN LET x
=x-sx: LET y=y-sy: LET p=p-15: I
F SCREEN$ (x,y)="." THEN LET c=c
-1
4110 IF ATTR (x,y)=113 THEN GO T
O 3150
4140 RETURN
5430 READ sx1,sy1,sx2,sy2: IF sx
1=100 THEN RESTORE 9000: GO TO 5
430
5435 GO SUB 3200
5436 PRINT AT x1,y1;a$;AT x2,y2;
b$;
5440 LET x1=x1+sx1: LET y1=y1+sy
1: LET x2=x2+sx2: LET y2=y2+sy2:
 LET a$=SCREEN$ (x1,y1): LET b$=
SCREEN$ (x2,y2)
5450 RETURN
5510 IF x=x1 AND y=y1 THEN GO TO
 3000
5515 IF x=x1+1 AND y=y1 THEN GO
TO 3000
5516 IF x=x1 AND y=y1+1 THEN GO
TO 3000
5520 IF x=x2 AND y=y2 THEN GO TO
 3000
5530 IF x=x3 AND y=y3 THEN GO TO
 3000
5540 RETURN
9000 DATA 0,-2,0,-1,0,-2,0,-1,0,
-2,0,-1,2,0,0,-1,0,-2,-1,0,-2,0,
-1,0,-2,0,0,-1,0,-2,0,-1,0,-2,0,
-1,0,-2,0,-1,0,-2,1,0,0,-2,1,0,0
,-2,1,0,0,-2,1,0,0,-2,0,-1,0,-2,
0,-1,0,-1,0,-1,2,0,0,-1,2,0,0,-1
,2,0,0,-1,2,0,0,-1,2,0,-1,0,0,2,
-1,0,-2,0,0,-1,-2,0,0,-1,-2,0,0,
-1,-2,0,0,-1
9001 DATA 0,2,1,0,0,2,1,0,0,2,0,
-1,0,2,0,-1,-2,0,-1,0,0,2,-1,0,0
,2,-1,0,0,2,-1,0,0,2,0,1,0,2,0,1
,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1
,0,1,1,0,2,0,1,0,2,0,0,1,2,0,0,1
,2,0,0,1,2,0,0,1,0,-2,0,1,0,-2,0
,1,0,-2,0,1,0,-2,-1,0,0,-2,-1,0,
-2,0,0,1,-2,0,0,1,0,1,2,0,0,1,0,
1,-2,0,0,1,-2,0,0,1,0,2,0,1,0,2,
0,1,0,2,0,1,100,0,0,0
```

Regata

Grazie all'avventura di Azzurra all'America's Cup, la vela è diventata improvvisamente uno sport molto conosciuto: anche se questo programma è stato concepito quando la più diffusa attività agonistica era la caccia ai dinosauri (e quella più diffusa tra i dinosauri la caccia ai bipedi spelacchiati), è certamente questo il momento più opportuno per la sua pubblicazione.

di **Claudio Lenzi**

Due componenti sono essenziali per ben figurare in regata: la prima è la preparazione fisica e tecnica dell'equipaggio e la messa a punto della barca per farla rendere al massimo nelle varie andature e nelle manovre, la seconda è la tattica ossia la scelta tempestiva della rotta da seguire e del momento in cui effettuare le manovre, in funzione delle mutevoli condizioni meteo e della posizione relativa alle altre barche; è proprio per esercitarsi su quest'ultimo aspetto della competizione che è stato scritto questo programma.

In una regata di triangolo olimpico si percorrono tre lati controvento ed altrettanti ad andatura portante; in questi ultimi però, la tattica è di minore importanza, in quanto ci si dirige direttamente verso la boa, perciò il percorso di regata è stato volutamente limitato al solo lato controvento.

Fatta l'ipotesi che le barche siano tutte sostanzialmente uguali, la velocità di navigazione dipende esclusivamente dall'andatura e cioè dall'angolo tra la rotta e la direzione del vento; è quindi stata calcolata una approssimazione analitica della polare sperimentale di velocità relativa a una barca a bulbo.

Dalla figura 1 si vede come non sia possibile dirigersi direttamente verso la boa al vento, in quanto le vele, fileggiando, non darebbero alcuna propulsione: è necessario quindi bolinare, ossia procedere a zig-zag, scegliendo di dirigersi ora a dritta ora a sinistra; non esiste inoltre un angolo ideale per risalire il vento, ma si deve di momento in momento scegliere se navigare di bolina stretta (40/45 o 320/315) compiendo un percorso breve, ma navigando lentamente, oppure di bolina larga (45/50 o 315/310) navigando più veloci su un percorso più lungo.

L'uso del programma

Dato il RUN, a richiesta, viene plottata la polare di velocità e stampata la tabella con l'assegnazione della tastiera, segue l'INPUT del numero dei timonieri (fino a sette); digitando 0 viene effettuata una breve regata dimostrativa con una

Figura 1 - La stampa ottenuta dall'esecuzione del programma Regata.

Seguito figura 1.

sola barca controllata dal calcolatore.
A questo punto il calcolatore chiede la lunghezza del percorso di regata (tra 0.5 e 2 miglia); consiglio caldamente le 2 miglia perché in questo modo le regate risultano più appassionanti e didatticamente più valide.
Viene ora disegnata la linea di partenza e la direzione del vento.
Le barche e i timonieri sono ora identificati con numeri da uno a sette, le barche sono sulla linea di partenza, in posizione da definire, già orientate a 45 o 315 gradi, a seconda quale direzione è più conveniente.

Da adesso in poi sono disponibili vari comandi, come si vede in figura 1; con le frecce verticali è possibile selezionare il timoniere interessato:
(T) per assegnare un nome al timoniere (al massimo quattordici caratteri), invece del numero; automaticamente il nome della barca sarà il primo carattere digitato, ma in caso di nomi con iniziale uguale si può cambiarlo con
(B) assegnando così alla barca un carattere qualsiasi.
È possibile, ovviamente, modificare la rotta; i co-

SHARP

Seguito Figura 1.

mandi disponibili a tale scopo sono:
- "(" mura a dritta (angolo al vento tra 30 e 180);
- ")" mura a sinistra (angolo al vento tra 320 e 180);
- "=" indica al calcolatore che si vuole impostare una nuova rotta (angolo col vento);
- 0...9 per porre l'angolo al vento tra 40 e 49 gradi;
- "L" per far impostare la rotta dal computer di bordo.

L'ultima operazione prima della partenza è il posizionamento della barca sulla linea, da eseguirsi premendo il tasto P; se la direzione del vento è inclinata rispetto alla linea di partenza, ad esempio a sinistra, chi parte da questa estremità è avvantaggiato, ma attenzione: un salto di vento è sempre in agguato.

Solo quando tutti i concorrenti sono in posizione si può partire premendo SPACE; la barca controllata dal calcolatore (già, anche il PC-1500 ha la sua barca che gareggia contro di noi ad armi pari e spesso dà del filo da torcere anche ai timonieri più smaliziati) sceglie una posizione tra quelle libere e la rotta più conveniente.

Durante il calcolo ed il plottaggio della nuova posizione di ogni barca appare sul display, sia la rotta scelta dal computer, in modo che ognuno possa confrontarla con la propria, sia la variazione di direzione che il vento subirà nel turno successivo, per aver modo di decidere per tempo la tattica da seguire.

Il plottaggio del vento a vettori compone una curva continua ed ogni tre turni di gioco viene tracciato un allineamento perpendicolare al vento, in modo da rendersi conto di come siano modificate le posizioni dei concorrenti.

Nei turni successivi non ha più importanza l'ordine in cui i timonieri immettono le eventuali variazioni di rotta; è anche possibile, prima di premere lo SPACE per cominciare il turno successivo, tor-

```
100:REM *********
102:REM *       *
104:REM * REGATA *
106:REM *       *
108:REM *   by   *
110:REM *C. LENZI*
112:REM * Milano *
114:REM *********
116:REM
118:TEXT :CSIZE 6:
    COLOR 1:LPRINT
    "REGATA";:TEXT
    :LPRINT :COLOR
    2:LPRINT "CLAU
    DIO LENZI (MI)
    ":LF 2
120:INPUT "Polare
    velocita ? ";R
    $:IF LEFT$ (R$
    ,1)<>"N"GOSUB
    "PV
122:INPUT "Assegna
    zione tastiera
     ? ";R$:IF
    LEFT$ (R$,1)<>
    "N"GOSUB "AT
124:GOTO "S
126:REM
128:REM VELOCITA'
    BARCA
130:REM
132:"VB"R=ABS R(T)
    :IF R<20STOP
134:IF R<55LET V(T
    )=IV*√(.64-1E-
    4*(R-100)^2):
    RETURN
136:IF R<115LET V(
    T)=IV*(R*6.783
    E-3+.29):
    RETURN
138:IF R<145LET V(
    T)=IV*(.85+√(.
    26*.26-1E-4*(R
    -130)^2)):
    RETURN
140:V(T)=IV*(1.6-√
    (.64*.64-1E-4*
    (R-180)^2)):
    RETURN
142:REM
144:REM   CALCOLO
    ROTTA
146:REM
148:"CR"DY=Y(T)-39
    7:IF DY=0LET D
    Y=.1
150:DX=X(T)-79:D=7
    /√(DX^2+DY^2):
    IF D>1GOTO "RE
    T
152:RB=ATN (DX/DY)
    :IF DY>0LET RB
    =RB-180*SGN DX
154:RB=RB+ASN D-V
156:DX=X(T)-138:D=
    10/√(DX^2+DY^2
    ):IF D>1GOTO "
    RET
158:RG=ATN (DX/DY)
    :IF DY>0LET RG
    =RG-180*SGN DX
160:RG=RG-ASN D-V
162:IF RB>-30AND R
    G<30GOTO "B
164:IF RB>-40AND R
    G<40LET R(T)=(
    45+P(T))*SGN (
    103-X(T))
166:IF RB>BSLET R(
    T)=RB:RETURN
168:IF RG>BSAND RB
    >-40LET R(T)=B
    S:RETURN
```

Figura 2 - Il listato del programma che permette di simulare una regata.

Seguito figura 2.

```
170:IF RG<BDLET R(
    T)=RG:RETURN
172:IF RB<BDAND RG
    <40LET R(T)=BD
    :RETURN
174:IF V<0GOTO "A
176:REM
178:REM ARRIVO IN
    BOA
180:REM
182:IF RB>-40LET R
    (T)=45+P(T):
    RETURN
184:IF RB>BDLET R(
    T)=RB:RETURN
186:R(T)=BD:RETURN
188:REM
190:REM ARRIVO IN
    GIURIA
192:REM
194:"A"IF RG<40LET
    R(T)=-45-P(T):
    RETURN
196:IF RG<BSLET R(
    T)=RG:RETURN
198:R(T)=BS:RETURN
200:REM
202:REM BOLINA
204:REM
206:"B"DX=X(T)-X0:
    DV=VM-V:DB=(RB
    +RG)/2
208:IF ABS DV<ABS
    (DX/15)LET R(T
    )=(45+P(T))*
    SGN -DX:RETURN
210:R(T)=(45+ABS D
    V/6)*SGN DV:
    RETURN
212:REM
214:REM *** LOOP
    PRINCIPALE **
216:REM
218:"P"BD=-45-V/(2
    .5-.5*SGN V):B
    S=45-V/(2.5+.5
    *SGN V):CSIZE
    1
220:WAIT 0:F=0:FOR
    T=0TO N:IF FR(
    T)=-1OR FR(T)=
    1GOTO "Q
222:F=F+1:IF FR(T)
    =2GOSUB "CR":
    GOSUB "VB
224:"Q"NEXT T:IF F
    =0GOTO "SC
226:IF FR(0)<2GOTO
    "L
228:IF N=0AND Y(0)
    =LLET P=1:GOTO
    "PAR
230:IF (F=1AND FR(
    0)=2)OR CO=1
    LET CO=0:GOTO
    "NAV
232:REM
234:REM INPUT DATI
     TIMONIERI
236:REM
238:"L"T=T+1:IF T>
    NLET T=1
240:"U"CLS :IF FR(
    T)=-1OR FR(T)=
    1GOTO "L
242:USING "+###":F
    =0:IF FR(T)=2
    AND L=0LET F=1
    :ON P+1GOTO "E
    ","D
244:R=R(T):IF R<0
    LET R=R+360
246:IF P=0GOTO "E
248:REM
```

Seguito figura 2.

```
250:REM DISPLAY IN
     REGATA
252:REM
254:"D"CURSOR 0:
    CPRINT "3C403C
    007C545C007C08
    78027F42007C44
    7C00";:PRINT V
    V;:GPRINT "000
    70507
256:CURSOR (18-LEN
    @$(T+1)):PRINT
    USING "####";@
    $(T+1);:IF F=0
    AND R(T)<=0
    PRINT R;"      "
    :CURSOR 23
258:IF F=0AND R(T)
    >0PRINT "      "
    ;R:CURSOR 19
260:IF F=1PRINT "
    *******":GOTO
    "INK
262:IF F=0AND FR(T
    )=2PRINT "***
264:GOTO "INK
266:REM
268:REM DISPLAY IN
     PREPARTENZA
270:REM
272:"E"X=(X(T)-78)
    /6
274:PRINT U$+V$;:
    CURSOR 18:
    PRINT STR$ X;:
    CURSOR 22:IF F
    R(T)=0PRINT
    USING "####";R
    ;
276:IF FR(T)=2
    PRINT " ***";
278:CURSOR 2:PRINT
    @$(T+1):CURSOR
    14:PRINT @$(T+
    11)
280:"ENT"CURSOR 21
    :PRINT "=:
282:REM
284:REM INKEY$ E I
    NPUT RELATIVI
286:REM
288:"INK"BEEP 1
290:"I"W$=INKEY$ :
    K=ASC W$:IF K=
    0GOTO "I
292:IF K=41LET R(T
    )=45+P(T):
    GOSUB "VB":FR(
    T)=0:GOTO "U
294:IF K=40LET R(T
    )=-45-P(T):
    GOSUB "VB":FR(
    T)=0:GOTO "U
296:"SU"IF K=10LET
    T=T-1:IF T<1
    LET T=N
298:IF FR(T)=-1OR
    FR(T)=1GOTO "S
    U
300:"GIU"IF K=11
    GOTO "L
302:IF K=32AND P=1
    CLS :GOTO "NAV
304:IF K=76AND FR(
    T)<2GOSUB "CR"
    :GOSUB "VB":FR
    (T)=2:GOTO "U
306:IF K=61GOTO "P
    R
308:IF K>47AND K<5
    8LET K=40+VAL
    W$:GOTO "PR1
310:IF K=86LET MP=
    MP-50:IF MP<-7
```

Seguito figura 2.

```
   5LET MP=-75
312:IF K=86
    GLCURSOR (XL,M
    P)
314:IF K=82CLS :
    PRINT @$(T+1);
    :CURSOR 10:
    INPUT " RET ?
    ";W$:IF LEFT$
    (W$,1)="S"CLS
    :GOTO "RET
316:IF K=84CLS :
    PRINT "Nome ti
    m.     ";@$(T+1)
    :CURSOR 10:
    INPUT @$(T+1):
    @$(T+11)=LEFT$
    (@$(T+1),1)
318:"N"IF K=66CLS
    :PRINT "Nome b
    arca   ";@$(T+1
    1):CURSOR 21:
    INPUT @$(T+11)
    :IF LEN @$(T+1
    1)>1GOTO "N
320:IF K=80AND P=0
    GOSUB "PP":
    GOTO "U
322:IF K=32AND P=0
    FOR I=1TO N:IF
    @$(T+1)<>""AND
    @$(T+11)<>""
    AND X(I)<>78
    NEXT I:P=1:
    GOTO "PAR
324:GOTO "U
326:REM
328:REM PARTENZA
330:REM
332:"PAR"CLS :
    PRINT "PARTENZ
    A":BEEP 1,100,
    1000:F=SGN V
334:IF SGN V<>SGN
    CVAND ABS V<
    ABS (CV*C)AND
    N>0LET F=SGN C
    V
336:X(0)=105+21*F:
    T=0:GOSUB "CR"
    :GOSUB "VB":IF
    N=0GOTO "M
338:"O"FOR T=1TO N
    :IF X(T)=X(0)
    LET X(0)=X(0)-
    6*F:GOTO "O
340:NEXT T
342:"M"FOR T=0TO N
    :GLCURSOR (X(T
    ),LP):LPRINT @
    $(T+11):NEXT T
    :C=0
344:REM
346:REM *** NAVIGA
    ZIONE ***
348:REM
350:"NAV"R=INT (R(
    0)+.5):IF R<0
    LET R=R+360
352:IF FR(0)=2
    CURSOR 17:
    PRINT USING "#
    ###";A$;R;
354:VP=V:V=V+CV+9-
    RND 17:IF ABS
    V>35LET V=(25+
    RND 10)*SGN V
356:VV=INT (V-VP+.
    5):SI=SI+V:CS=
    CS+1:VM=SI/CS/
    1.5:MP=MI
358:CURSOR 0:PRINT
    USING "+###";"
    nuovo ril.";VV
```

Seguito figura 2.

```
    ;:GPRINT "0007
    0507
360:REM
362:REM  COLORE E
    VENTO MEDIO
364:REM
366:IF N=0OR AR=1
    GOTO "R
368:C=C+.25:IF C=3
    LET C=0
370:COLOR 3:A=0:IF
    C-INT C=0AND M
    A<394GLCURSOR
    (0,Y(0)):COLOR
    C:A=1:SI=VM*12
    :CS=8:SX=0:SY=
    0
372:REM
374:REM  FOR/NEXT
    DI NAVIGAZIONE
376:REM
378:"R"AV=AV+1:CM=
    0:FOR T=0TO N:
    IF FR(T)=-1OR
    FR(T)=1GOTO "N
    EXT
380:REM
382:REM POSIZIONE
384:REM
386:RC=R(T)+VP:X(T
    )=X(T)+V(T)*
    SIN RC:Y(T)=Y(
    T)+V(T)*COS RC
388:REM
390:REM MAX E MIN
392:REM
394:IF CM=0LET MI=
    Y(T):MA=MI:MS=
    X(T):MD=MS:
    GOTO "T
396:IF Y(T)<MILET
    MI=Y(T)
398:IF Y(T)>MALET
    MA=Y(T)
400:IF X(T)<MSLET
    MS=X(T)
402:IF X(T)>MDLET
    MD=X(T)
404:REM
406:REM STAMPA BAR
    CHE
408:REM E VERIFICA
     ARRIVO
410:REM
412:"T"IF X(T)>-5
    AND X(T)<216
    GLCURSOR (X(T)
    ,Y(T)):LPRINT
    @$(T+11)
414:IF Y(T)>350LET
    FR(T)=2
416:IF A=1LET SX=S
    X+X(T):SY=SY+Y
    (T)
418:CM=CM+1:IF Y(T
    )>394GOSUB "CL
420:"NEXT"NEXT T:
    CURSOR 17:
    PRINT "
       ":IF A=0OR A
    R=1GOTO "V
422:X=SX/CM:Y=SY/C
    M
424:LINE (MS-10,Y+
    3+(X-MS+10)*
    TAN VP)-(MD+15
    ,Y+3-(MD+15-X)
    *TAN VP)
426:REM
428:REM STAMPA VEN
    TO
430:REM
432:"V"IF AR=1GOTO
    "F
```

Seguito figura 2.

```
434:XF=XI+(MA-YI)*
    SIN UP
436:IF XI<40LET XL
    =XF+180:LINE (
    XI+180,YI)-(XL
    ,MA)
438:IF XI>0LINE (X
    F,MA)-(XI,YI):
    XL=XI
440:XI=XF:YI=MA
442:"F"IF MA>200
    AND L<400
    GLCURSOR (0,MA
    ):COLOR 0:L=40
    0:GOSUB "LINEA
    "
444:W$=INKEY$ :IF
    W$=" "OR N=0
    BEEP 1:CO=1:
    GOTO "P
446:MP=MP-65:IF MP
    <-75LET MP=-75
448:GLCURSOR (XL,M
    P):GOTO "P
450:REM
452:REM FINE LOOP
    PRINCIPALE
454:REM
456:REM DISEGNO LI
    NEA
458:REM
460:"LINEA"
    GLCURSOR (79,-
    2+L):LPRINT "o
    ":LINE (84,L)-
    (134,L)
462:LINE (134,L-2)
    -(134,L+2)-(13
    6,L+6)-(138,L+
    8)
464:LINE -(140,L+6
    )-(142,L+2)-(1
    42,L-2)-(141,L
    -4)-(139,L-6)
466:LINE -(137,L-6
    )-(135,L-4)-(1
    34,L-2):RETURN
468:REM
470:REM POSIZIONE
    DI PARTENZA
472:REM
474:"PP"K=0:CLS :
    PRINT "Pos.par
    t.(1/8)=";(X(T
    )-78)/6:CURSOR
    22:INPUT K
476:K=INT K:IF K<1
    OR K>8GOTO "PP
478:X(T)=INT (K*6+
    78):FOR I=1TO
    N:IF T=IOR X(T
    )<>X(I)NEXT I:
    RETURN
480:BEEP 3:X(T)=78
    :GOTO "PP
482:REM
484:REM PRUA
486:REM
488:"PR"K=45+P(T):
    CLS :PRINT "Pr
    ua "+@$(T+1)+"
     ("+STR$ (45+P
    (T))+") = ";:
    INPUT K
490:IF K<32OR K>18
    0GOTO "PR
492:"PR1"P(T)=K-45
    :FR(T)=0:R(T)=
    K*SGN R(T)
494:GOSUB "VB":
    GOTO "U
496:REM
498:REM RITIRO
500:REM
```

Seguito figura 2.

```
502:"RET"FOR J=NTO
    0STEP -1:IF CL
    (J)<>0NEXT J
504:FR(T)=-1:X(T)=
    T*6:CL(J)=T/10
    :IF N=0GOTO "S
    C
506:IF CL(J-1)=0
    GOTO "P
508:GOTO "SC
510:REM
512:REM CLASSIFICA
514:REM
516:"CL"IF Y(T)-39
    6<76-X(T)OR Y(
    T)-394<X(T)-13
    8RETURN
518:BEEP 1:K=INT A
    V*1E5+(490-INT
    Y(T))*1E3+INT
    (200-X(T))+T/1
    0:FR(T)=1:J=0:
    F=0:AR=1
520:"G"IF CL(J)=0
    LET F=1:CL(J)=
    K:IF J=NGOTO "
    SC
522:IF F=1AND CL(J
    +1)<>0GOTO "SC
524:IF F=1RETURN
526:IF INT K<INT C
    L(J)LET Q=CL(J
    ):CL(J)=K:K=Q:
    GOTO "H
528:IF INT K>=INT
    CL(J)GOTO "H
530:IF CL(J+1)<>0
    GOTO "SC
532:"H"J=J+1:IF J<
    =NGOTO "G
534:REM
536:REM STAMPA CLA
    SSIFICA
538:REM
540:"SC"USING :
    GLCURSOR (0,LP
    -40):TEXT :
    CSIZE 1:COLOR
    1
542:IF N=0BEEP ON
    :LF 6:GOTO "S
544:RESTORE :FOR I
    =0TO N+1:V(I)=
    I+1:READ X(I):
    NEXT I:X(N+1)=
    X(N)
546:DATA 0,3,5.7,8
    ,10,11.7,13,14
    ,15,16
548:REM
550:FOR I=0TO N:IF
    I=NGOTO "W
552:IF INT CL(I)=
    INT CL(I+1)LET
    V(I+1)=V(I)
554:"W"T=(CL(I)-
    INT CL(I))*10
556:V=V(I):IF FR(T
    )=-1LET V=N+2
558:CF(T)=CF(T)+(V
    )*10^(PR-1)
560:MA=(CF(T)-INT
    CF(T))*10
562:IF V>MALET CF(
    T)=CF(T)+(V-MA
    )/10
564:IF PR=4LET CF(
    T)=CF(T)-X(MA-
    1)*1E7
566:IF PR<4OR V<=M
    ALET CF(T)=CF(
    T)+X(V-1)*1E7
568:IF PR>3AND V>M
    ALET CF(T)=CF(
```

Seguito figura 2.

```
     T)+X(MA-1)*1E7
570:NEXT I
572:LPRINT "*** CL
     ASSIFICA DOPO
     LA";:FOR I=0TO
     N:R(I)=INT CF(
     I)+I/10:NEXT I
574:LPRINT PR;" PR
     OVA  ***":
     LPRINT
576:REM
578:REM SORT
580:I=0
582:"X"FOR J=I+1TO
     N:IF INT R(I)<
     =INT R(J)GOTO
     "Y
584:R=R(I):R(I)=R(
     J):R(J)=R
586:"Y"NEXT J:I=I+
     1:IF I<NGOTO "
     X
588:REM
590:FOR I=0TO N:T=
     (R(I)-INT R(I)
     )*10:REM R=INT
     (CF(T)/10^(6-J
     ))/10:R=(R-INT
     R)*10
592:LPRINT STR$ (I
     +1)+" "+@$(T+1
     1)+" "+@$(T+1)
     ;
594:FOR J=1TO PR:
     TAB (17+J*2)
596:R=INT (CF(T)/1
     0^(J-1))/10:R=
     (R-INT R)*10:W
     $=STR$ R:IF R=
     N+2LET W$="R
598:LPRINT W$;:
     NEXT J
600:TAB 31:K=INT (
     CF(T)/1E6)/10:
     IF K<10LPRINT
     " ";
602:LPRINT K:NEXT
     I:LF 6:IF PR<6
     GOTO "PROVA
604:LF 5:END
606:REM
608:REM POLARE VEL
     OCITA"
610:REM
612:"PV"CSIZE 2:
     GRAPH :COLOR 3
     :GLCURSOR (20,
     0):LPRINT "Pol
     are velocita":
     LINE (201,7)-(
     203,11)
614:GLCURSOR (0,-1
     20):SORGN
616:LINE (0,0)-(21
     4,0),0,0
618:GLCURSOR (16,7
     0):LPRINT "315
     ":GLCURSOR (17
     4,70):LPRINT "
     45"
620:GLCURSOR (104,
     40):ROTATE 3:
     LPRINT "vento":
     LINE (107,100
     )-(107,25)-(11
     0,35)-(104,35)
     -(107,25)
622:LINE (107,0)-(
     107,-65):
     GLCURSOR (90,-
     80):ROTATE 0:
     LPRINT "180":
     LINE (107,-83)
     -(107,-107)
```

Seguito figura 2.

```
624:CLCURSOR (0,-1
   07):LINE (0,-1
   07)-(172,65),2
   ,1:LINE (194,8
   7)-(214,107)
626:LINE (214,-107
   )-(42,65),2,1:
   LINE (20,87)-(
   0,107)
628:LINE (0,107)-(
   214,-107),0,,B
630:COLOR 3:
   CLCURSOR (107,
   0):SORGN
632:CLEAR :DIM R(0
   ),V(0):IV=100
634:FOR I=24TO 336
   STEP 8:R(0)=I:
   IF I>180LET R(
   0)=360-I
636:GOSUB "VB":
   LINE -(V(0)*
   SIN I,V(0)*COS
   I)
638:NEXT I:LINE -(
   0,0):TEXT :LF
   8:RETURN
640:REM
642:REM ASSEGNAZIO
   NE TASTIERA
644:REM
646:"AT"TEXT :
   CSIZE 1:COLOR
   3:LPRINT "* *
   *  ASSEGNAZION
   E TASTIERA  *
   * *
648:COLOR 0:LPRINT
   :LPRINT "T : N
   ome del timoni
   ere":LPRINT "B
   : Nome della
   barca
650:LPRINT "P : Po
   sizione di par
   tenza
652:COLOR 1:LPRINT
   :LPRINT "( : M
   ure a dritta":
   LPRINT ") : Mu
   re a sinistra
654:LPRINT "= : An
   golo col vento
    (30/180)
656:LPRINT "0...9
   : Angolo col v
   ento (40/49)":
   LPRINT "L : Ro
   tta scelta dal
    computer
658:COLOR 2:LPRINT
   :LPRINT "V : P
   er rivedere il
    percorso
660:LPRINT "R : Ri
   tiro timoniere
662:LPRINT "Frecci
   e vert. : Rota
   zione timonier
   e
664:LPRINT "SPACE
   : Partenza e n
   avigazione":LF
   6:RETURN
666:REM
668:REM START SENZ
   A ISTRUZIONI
670:REM
672:"S"CLEAR :BEEP
   ON :WAIT 0:
   INPUT "N. TIMO
   NIERI (0=DIM.)
    ? ";N
674:N=INT N:IF N<0
```

Seguito figura 2.

```
     OR N>7GOTO "S
676:IF N=0PRINT "D
     IMOSTRAZIONE:
     Bol.= .5 MM
678:"DIM"DIM P(N),
     R(N),V(N+1),X(
     N+1),Y(N),FR(N
     ),CL(N),CF(N)
680:IV=25+N*5:IF N
     =0LET IV=50:
     BEEP OFF
682:FOR T=1TO N:@$
     (T+1)=CHR$ (T+
     48):@$(T+11)=@
     $(T+1):NEXT T
684:TEXT :CSIZE 1:
     COLOR 0:LF 2
686:REM
688:REM INIZIO PRO
     VA
690:REM
692:"PROVA"WAIT 0:
     L=.5+1.5*SGN N
     :IF N>0INPUT "
     LATO BOLINA (.
     5-2 MM) ? ";L
694:IF L<.5OR L>2
     GOTO "PROVA
696:L=400-200*L:LP
     =L:C=(29-N*2)*
     (1-L/400):CV=(
     31-RND 61)/C
698:P=0:MI=L:AR=0:
     CO=0:IF N>0LET
     PR=PR+1
700:GRAPH :CSIZE 1
     :GLCURSOR (0,-
     370):SORGN :
     GOSUB "LINEA
702:V=8.5-RND 15:X
     I=20:YI=L
704:SI=V*8:CS=8:VM
     =V/1.5:X0=107+
     CV*100*(1-L/40
     0)
706:X=30*SIN V:Y=3
     0*COS V:LINE (
     200-X,L-Y)-(20
     0,L)
708:GLCURSOR (127,
     L-8):LPRINT "8
     ":GLCURSOR (87
     ,L-8):LPRINT "
     1
710:LINE (20,L)-(2
     0-X,L-Y)
712:GLCURSOR (0,L-
     75):COLOR 3
714:U$="T:
        ":V$="B:   P:
     ":IF N>0PRINT
     U$+V$
716:A$="SHARP":K$=
     "#":FR(0)=2
718:FOR T=0TO N:X(
     T)=78:Y(T)=L:F
     R(T)=0:CL(T)=0
720:R(T)=(45+P(T))
     *SGN -V:GOSUB
     "VB":NEXT T:FR
     (0)=2:X(0)=107
     :GOTO "P
65100:REM
65110:REM RENUMBER
65120:REM
65130:"REN"N=100:I
       =2:M=STATUS
       2-STATUS 1
65140:H=N/256:P=
       INT H:S=(H-P
       )*256:POKE M
       ,P,S:N=N+I:M
       =M+3+PEEK (M
       +2):IF PEEK
       M<254GOTO 65
       140

STATUS 1
               8847

STATUS 3-STATUS 2
                102
```

nare sulle proprie decisioni, modificandole.

Durante la regata bisogna astenersi nel modo più assoluto dal pigiare il tasto per l'avanzamento della carta (per i più distratti consiglio di coprire il tasto della stampante con un coperchietto); per rivedere il percoso effettuato tenete invece premuto V.

Nel caso che un concorrente voglia assentarsi momentaneamente, oppure se ci si vuole cimentare contro più barche, è possibile in qualsiasi momento far controllare una o più barche dal calcolatore pigiando L (questo è particolarmente utile nell'ultima parte della regata, quando si tratta di convergere verso la linea di arrivo): per riprendere poi il controllo basta inserire la rotta voluta.

È inoltre possibile ritirarsi pigiando R e rispondendo affermativamente alla richiesta di conferma.

Quando tutte le barche hanno terminato la regata viene stilata la classifica secondo il sistema di punteggio olimpico, che assegna 0 punti al primo e nell'ordine 3, 5.7, 8, 10, 11.7, 13, 14, 15, ecc.

Tutti i ritirati prendono il punteggio spettante all'ultimo (se fossero arrivati tutti). Dopo ogni prova vengono sommati i punteggi ottenuti da ciascun concorrente ed aggiornata la classifica. Se si effettuano più di tre prove, la classifica viene stilata senza tenere conto del piazzamento peggiore; perciò, se in una delle prove si è andati male o ci si è ritirati, si può sempre contare sullo scarto. In caso di parità prevale chi ha meglio figurato nell'ultima prova svolta.

Il programma occupa oltre 8 Kbyte ed è ovviamente necessario che il PC-1500 sia in configurazione espansa. Nel caso vi siano otto barche in regata la memoria non utilizzata (STATUS 3 - STATUS 2) si riduce a meno di cento byte: è consigliabile perciò digitare NEW 0 prima di caricare il programma.

Se avete delle routine in linguaggio macchina nell'area protetta, potrete utilizzare il programma con un numero minore di timonieri. La numerazione ha incremento due per mantenere tutti i numeri di linea inferiori a mille, ottimizzando così la stampa del listato; chi volesse può facilmente rinumerarlo, grazie alla routine REN posta in coda al programma.

Qualche regatante incallito commenterà che il percorso non è completo, che non si tiene conto delle eventuali collisioni fra le imbarcazioni e delle variazioni di intensità del vento e della corrente, magari differenziate in varie parti del campo di regata, ma dopo un lungo rodaggio del programma ho ritenuto che quello scelto fosse, tutto sommato, un buon compromesso fra simulazione, divertimento e didattica, considerando anche la relativamente limitata disponibilità di memoria ed i pesanti tempi di elaborazione.

Buon vento...

L'ing. Sebastiano Zappalà, autore di "Magicatalog" pubblicato sul n. 51, consiglia una nuova release del suo programma al quale, quindi, possono essere apportate le seguenti modifiche:

```
0   REM ==>> JUMBO UTILITIES <<==

310  VTAB 13: GOSUB 1000: GOSUB 9000:N = 0: VTAB 13: PRINT D$OP$FI$: PRINT
     D$RE$FI$: ONERR  GOTO 330

2000  HOME : GOSUB 9000: VTAB 1: HTAB 4: INVERSE : PRINT " MAGICATALOG ";:
      NORMAL : PRINT "----": VTAB 10: PRINT  CHR$ (4)"RUNMAGICATALOG"
```

e vanno abolite le righe 2010 e 2020.

```
0   REM ==>> MC2 <<==

210 S = S + (N = 7):N = N + 1 - N * (N = 7): ON S > K5 GOTO 220:CP = PS +
    K2 + K3 * N + K4 * S:PC =  PEEK (CP - 3): ON  NOT PC GOTO 220: ON PC =
    K0 GOTO 210:M1 = CP:M2 = CP + 29:M3 = CB: GOSUB 8000:CB = CB + 35: FOR
    I = 0 TO 4: POKE CB - 5 + I,V(I): NEXT : GOTO 210

1110  POKE PP,0: WAIT PX,128:P =  PEEK (PX): ON P <  > 141 AND P <  > 155
    GOTO 1110: POKE PP,0: ON P = 155 GOTO 470: ON J GOTO 10,430: STOP
```

La titolazione col computer

Questo programma è nato dall'esigenza di alcuni studenti di un liceo scientifico di simulare una titolazione. Da essi è stato sviluppato l'algoritmo generale di calcolo per la titolazione di acidi e basi di qualsiasi volume e concentrazione.

di **Paolo Coretti**

La struttura è lineare e molto semplice; dopo la visualizzazione sul display del menu principale (linee 5-85), si opera la scelta delle opzioni disponibili, che sono:
1 - calcolo del pH di una soluzione di acido forte (righe 100-102);
2 - calcolo del pH di una soluzione di base forte (righe 105-111);
3 - calcolo del pH di una soluzione di acido debole (righe 120-121);
4 - calcolo del pH di una soluzione di base debole (righe 130-131);
5 - calcolo del pH di una soluzione con idrolisi (righe 150-221);
6 - simulazione della titolazione di una soluzione di acido debole con una base forte (righe 250-390);
7 - calcolo del pH di una soluzione formata da un certo volume ed una data concentrazione di un acido debole con una base di un certo volume e di concentrazione data (righe 390-410).
Per il calcolo di pH di acidi e basi forti c'è un controllo per verificare se la concentrazione è diluita (meno di 10^{-6}M): si introduce una formula correttiva, che tiene conto degli idrogenioni dell'acqua, per evitare grossolani errori (pH = 8 per una soluzione 10^{-8}M di HC1). Le formule usate nelle parti 1,2,3,4,5 sono facilmente reperibili su qualsiasi testo di chimica. La sesta e settima parte del programma utilizzano gli algoritmi di calcolo riportati a parte, assieme al relativo flowchart per il calcolo del pH della titolazione. Quando il volume della base aggiunta è superiore a tre volte il volume iniziale dell'acido il pH, ormai asintotico, viene calcolato dalla formuletta della riga 390.

L'esecuzione è semplicissima: basta dare il RUN ed immettere i dati richiesti dal calcolatore.
La concentrazione dell'acido è bene non sia troppo bassa ($10^{+1} - 10^{-3}$M) perché altrimenti al pH iniziale, e nei punti di equivalenza, c'è l'interferenza degli idrogenioni dell'acqua ed in questi casi bisognerebbe introdurre algoritmi di calcolo alquanto più complicati.

Algoritmi dei calcoli

1) $C_a = \dfrac{V_1 N_1}{V_1 + V_2}$ $\quad C_b = \dfrac{V_2 N_2}{V_1 + V_2}$

C_a = concentrazione.
C_b = concentrazione base

V_1 = volume acido all'inizio della titolazione.
V_2 = volume base all'inizio della titolazione.
N_1 = concentrazione acido.
N_2 = concentrazione base.

2) Soluzione tampone:

$$C_a = \frac{V_1 N_1 - V_2 N_2}{V_1 + V_2} \qquad C_b = \frac{V_2 N_2}{V_1 + V_2}$$

$$[H^+] = K_a \frac{C_a}{C_b} \qquad [H^+] = K_a \frac{V_1 N_1 - V_2 N_2}{V_2 N_2}$$

Figura 1 - Un esempio d'esecuzione.

```
READY P0
RUN

A.F.(1)B.F.(2)A.D.(3
)B.D.(4)IDROLISI(5)T
ITOLAZIONE(6)ANALISI
(7)?
6
TIT.AC.DEB.-B.FORTE

V.AC.=?
100
C.AC=?
1E-2
C.BASE=?
2E-2
KA=?
1.8E-5
DV=?
10
PH INIZIALE= 3.37

V= 10.00PH= 4.14

V= 20.00PH= 4.57

V= 30.00PH= 4.92

V= 40.00PH= 5.35

V= 50.00PH IDROLISI=
 8.28

V= 10.00PH= 11.10

V= 20.00PH= 11.37

V= 30.00PH= 11.52

V= 40.00PH= 11.62

V= 50.00PH= 11.70

V= 60.00PH= 11.76

V= 70.00PH= 11.80

V= 80.00PH= 11.84

V= 90.00PH= 11.88

V= 100.00PH= 11.90

PH ASINTOTICO= 12.00
```

Figura 2 - Flow-chart per la simulazione della titolazione.

V_1 = volume iniziale acido - volume base aggiunta.
V_2 = volume base dopo le varie aggiunte.
$B = V_1 N_1 - V_2 N_2$. Se $B = 0$, si avrà l'idrolisi, e la concentrazione sarà:

$$3) \quad \frac{V_2 N_2}{V_1 + V_2}$$

in cui V_2 è il volume della base aggiunta nel momento dell'idrolisi.
Superato il punto di idrolisi, avremo:

$$4) \quad [OH^-] = \frac{V N}{(V_a + V_b) + V}$$

V = volume base aggiunta.
N = concentrazione base.
V_a = volume acido all'idrolisi.
V_b = volume base all'idrolisi.

Listato 1 - Il programma per lo studio del PH.

```
  5 VAC :SET N
 10 PRT "A.F.(1)B.F
    .(2)A.D.(3)B.D.
    (4)";
 20 PRT "IDROLISI(5
    )TITOLAZIONE(6)
    ";
 30 PRT "ANALISI(7)
    ";:INP A$:SET N
 35 IF A$="1" THEN
    100
 40 IF A$="2" THEN
    110
 50 IF A$="3" THEN
    120
 60 IF A$="4" THEN
    130
 70 IF A$="5" THEN
    150
 80 IF A$="6" THEN
    250
 85 IF A$="7" THEN
    400
100 INP "C=",C:IF C
    ≥1E-6 THEN 102
101 H=-LOG (((C+SQR
     (C*C+4*1E-14))
    /2)):PRT "PH=";
    H:GOTO 5
102 H=-LOG C:PRT "P
    H=";H:GOTO 5
105 H=14+LOG C:PRT
    "PH=";H:GOTO 5
110 INP "C=",C:IF C
    ≥1E-6 THEN 105
111 H=14+LOG ((C+SQ
    R (C*C+4*1E-14)
    )/2):PRT "PH=";
    H:GOTO 5
120 INP "C=",C:INP
    "K=",K:H=-LOG S
    QR (K*C):PRT "P
    H=";H
121 GOTO 5
130 INP "C=",C:INP
    "K=",K:H=-(LOG
    SQR (K*C))+14:P
    RT "PH=",H
131 GOTO 5
150 PRT "KA O KB(A/
    B)?";
160 B$=KEY:IF B$=""
     THEN 160
161 PRT
170 IF B$="A" THEN
    220
180 IF B$="B" THEN
    200
181 GOTO 160
200 INP "K=",K:INP
    "C=",C:H=-LOG S
    QR ((1E-14/K)*C
    )
201 PRT "PH=";H:GOT
    O 5
220 INP "K=",K:INP
    "C=",C:H=-(-LOG
     SQR ((1E-14/K)
    *C))+14
221 PRT "PH=";H:GOT
    O 5
250 VAC :PRT "TIT.A
    C.DEB.-B.FORTE"
260 INP "V.AC.=",H:
    INP "C.AC=",C
270 INP "C.BASE=",E
271 INP "KA=",K
272 INP "DV=",D
275 SET F2
280 Z=-LOG SQR (K*C
    ):PRT "PH INIZI
    ALE=";Z
290 A=0
300 A=A+D
305 B=H*C-A*E
310 IF B≤0 THEN 330
315 P=-LOG (K*B/(A*
    E))
320 PRT "V=";A;"PH=
    ";P
325 GOTO 300
330 Q=(A*E)/(H+A)
340 P=14+LOG SQR (1
    E-14/K*Q)
350 PRT "V=";A;"PH
    IDROLISI=";P
355 I=A+H:A=0
360 A=A+D
365 P=14+LOG ((A*E)
    /(I+A)):SET F2
370 PRT "V=";A;"PH=
    ";P
380 IF A≤H-D THEN 3
    60
390 G=(INT P)+1:PRT
     "PH ASINTOTICO
    =";G:SET N:END
400 INP "V.AC.=",V:
    INP "C.AC.=",C:
    INP "KA=",K:SET
     N
401 INP "V.BASE=",D
    :INP "C.BASE=",
    E
402 Q=V*C-(D*E)
403 IF Q>0 THEN 406
404 IF Q=0 THEN 407
405 IF Q<0 THEN 410
406 H=-LOG ((K*Q)/(
    D*E):SET F2:PRT
     "TAMPONE:PH=";
    H:GOTO 400
407 H=14+LOG SQR ((
    1E-14/K)*((D*E)
    /(V+D)):SET F2
408 PRT "IDROLISI:P
    H=";H:GOTO 400
410 H=(ABS Q)/(V+D)
    :H=14+LOG H:PRT
     "PH=";H:GOTO 4
    00
```

Infine, per il calcolo dell'ultima opzione, avremo, nel caso di base in eccesso rispetto all'acido:

$$5) \quad [OH^-] = \frac{|V_1 N_1 - V_2 N_2|}{V_1 + V_2}$$

PL/Bit: il compilatore

di **Carlo Magnaghi**

Parte prima

Nel suo libro "Algorithms+Data Structures=Programs", Niklaus Wirth (creatore del Pascal e, più recentemente, del Modula-2) descrive un linguaggio di programmazione ridotto all'osso, ma che del Pascal mantiene ancora l'aroma: il PL/0. Così com'è questo linguaggio non serve a molto (si tratta in effetti, più che altro, di un esempio didattico): manca infatti la possibilità di utilizzare qualunque tipo di variabile complessa come gli array e, ciò che più conta, non è possibile gestire l'input/output del calcolatore.
Con un po' di pazienza è tuttavia possibile modificare questo linguaggio, per ottenere qualcosa di sufficientemente completo: è nato così PL/Bit, un compilatore che, nella sua semplicità, permette di scrivere programmi abbastanza complessi (si vedano gli esempi Hilbert e Hanoi), generando un codice molto efficiente (ad esempio, nell'esecuzione del programma Hilbert, nonostante il grande numero di moltiplicazioni e divisioni eseguite per ottenere un disegno più corretto, il codice generato da questo compilatore è circa 5-6 volte più veloce di quello generato dall'Apple Pascal).

Listato 1 - Programma Hanoi. Questo listato, oltre ad illustrare l'utilizzo della ricorsione e dei vettori in PL/Bit, mostra un semplice esempio di utilizzo del comando α.

```
INTERFACE
 USEPAGE 1,
 SHOWPAGE 1,
 FILLSCREEN 1,
 HCOLOR 1,
 HPLOT 2,
 HPLOTTO 2;

CONST
 YMOVE=140;

VAR NUM,I,J;

ARRAY
 XMID 1..20,
 WIDTH 1..20,
 YLOW 1..20;

PROCEDURE MOVEDISK ORG,DST;

VAR
 XO,YO,XD,YD,X1,X2,XM,I,DISK,D;

BEGIN
 XO:=ORG*90+50;XD:=DST*90+50;
 YO:=200;YD:=190;
 I:=0;REPEAT
  I:=I+1;
  IF YLOW(I)<YO AND XMID(I)=XO THEN
   BEGIN
```

Seguito listato 1.

```
     YO:=YLOW(I);
     DISK:=I
    END;
   IF YLOW(I)<YD AND XMID(I)=XD THEN
    YD:=YLOW(I)
  UNTIL I=NUM;YD:=YD-5;
  XM:=XMID(DISK);
  X1:=XM-WIDTH(DISK);
  X2:=XM+WIDTH(DISK);
  REPEAT
   CALL HCOLOR 0;
   CALL HPLOT X1,YO;
   CALL HPLOTTO X2,YO;
   CALL HCOLOR 3;
   CALL HPLOT X1,YO-4;
   CALL HPLOTTO X2,YO-4;
   YO:=YO-1
  UNTIL YO<YMOVE;
  D:=2*(DST>ORG)-1;
  IF D<O THEN
   BEGIN I:=X1;X1:=X2;X2:=I END;
  REPEAT
   CALL HCOLOR O;
   CALL HPLOT X1,YO;
   CALL HPLOTTO X1,YO-3;
   X1:=X1+D;X2:=X2+D;XM:=XM+D;
   CALL HCOLOR 3;
   CALL HPLOT X2,YO;
   CALL HPLOTTO X2,YO-3;
  UNTIL XM=XD;
  REPEAT
   YO:=YO+1;
   CALL HCOLOR O;
   CALL HPLOT X1,YO-4;
   CALL HPLOTTO X2,YO-4;
   CALL HCOLOR 3;
   CALL HPLOT X1,YO;
   CALL HPLOTTO X2,YO;
  UNTIL YO=YD;
  XMID(DISK):=XD;
  YLOW(DISK):=YD
 END;

 PROCEDURE MOVE ORG,VIA,DST,NUM;

 IF NUM >O THEN
  BEGIN
   CALL MOVE ORG,DST,VIA,NUM-1;
   CALL MOVEDISK ORG,DST;
   CALL MOVE VIA,ORG,DST,NUM-1;
  END;
```

APPLE

Seguito listato 1.

```
REPEAT
@INLOOP LDA $C000
@ STA $C010
@ SEC
@ SBC #$B0
@ CMP #10
@ BCS INLOOP
@ CMP #0
@ BEQ INLOOP
@ STA VARSTACK
@ LDA #0
@ STA VARSTACK+1
 I:=0;REPEAT I:=I+1;
  XMID(I):=50;
  WIDTH(I):=5*(NUM+1-I);
  YLOW(I):=190-5*I
 UNTIL I=NUM;
 CALL USEPAGE 1;
 CALL FILLSCREEN 0;
 CALL SHOWPAGE 1;CALL HCOLOR 3;
 I:=191;REPEAT
  CALL HPLOT 0,I;
  CALL HPLOTTO 279,I;
  I:=I-1;
 UNTIL I=186;
 CALL HPLOTTO 279,0;
 CALL HPLOTTO 0,0;
 CALL HPLOTTO 0,191;
 I:=0;REPEAT I:=I+1;
  J:=0;REPEAT
   CALL HPLOT 50-WIDTH(I),YLOW(I)-J;
   CALL HPLOTTO 50+WIDTH(I),YLOW(I)-J;
   J:=J+1
  UNTIL J=4
 UNTIL I=NUM;
 CALL MOVE 0,1,2,NUM;
UNTIL 0.
```

Purtroppo non mi è possibile rimandare a manuali introduttivi su questo linguaggio: anche nel libro sopra citato si rimanda a una generica conoscenza del Pascal, cui il PL/0 assomiglia molto, dando solo un breve programma di esempio.
Questa prima parte dell'articolo sarà quindi dedicata a un'introduzione di tipo 'tutorial' a questa versione del linguaggio (che differisce in modo abbastanza sostanziale dal PL/0 standard).
I tipi di variabili usati in questo linguaggio sono interi e array monodimensionali di interi. A differenza da ciò che succede in BASIC, le variabili devono essere dichiarate prima dell'uso. Le dichiarazioni avvengono all'inizio del programma o della procedura (parlerò delle procedure più tardi). Per dichiarare le variabili intere (per esempio A e B) basta scrivere:

```
VAR A,B;
```

(da notare subito che il compilatore è case-sensitive, cioè riconosce come differenti le lettere maiuscole e minuscole, quindi "a" e "A" sono variabili diverse, mentre 'var', a differenza di 'VAR,' non ha nessun significato particolare per il compilatore).
La lista di variabili può essere lunga a piacere, i nomi devono essere separati da virgole e l'ultimo deve essere seguito da un punto e virgola.
Leggermente più complessa è la dichiarazione degli array. A differenza del BASIC, è possibile avere array con indice minimo diverso da zero. Una tipica dichiarazione è come segue:

```
ARRAY
```

```
A 0..8,
B 5..24,
C -6..82;
```

Da notare che il compilatore non riconosce alcuna differenza tra spazi e caratteri RETURN; una dichiarazione del tutto equivalente è la seguente:

```
ARRAY A 0.. 8,B
 5..24,C -6 .. 82

;
```

(Ovviamente la prima scrittura è più leggibile). Per comodità e chiarezza è pure possibile dichiarare delle costanti, per esempio:

```
CONST
 APPLES=2,
 WORMS=5;
```

L'utilità è in termini di leggibilità del programma e di facilità di modifica; supponiamo di voler modificare il programma e che ora le mele debbano essere 8: basterà modificare la dichiarazione iniziale; se invece avessimo scritto '2' al posto di 'APPLES' all'interno del programma dovremmo sostituire un sacco di caratteri, stando ben attenti a non cambiare in '8' anche dei '2', che con le mele non hanno niente a che fare.
Le dichiarazioni CONST, VAR, ARRAY devono apparire in questo ordine all'inizio del programma; consideriamo, per chiarire le idee, il seguente esempio:

```
CONST MAX1=50;
VAR I;
ARRAY NUM 0..MAX1;

BEGIN
 I:=0;REPEAT
  NUM(I):=0;
  I:=I+1
 UNTIL I>MAX1
END.
```

Anche se il significato del programma è abbastanza intuitivo, non preoccupatevi se per ora avete qualche difficoltà a leggere un linguaggio così diverso dal BASIC; ciò che invece dovrebbe già essere chiaro è che MAX1 è una costante di valore 50, I è una variabile semplice e NUM è un array il cui indice varia tra 0 e MAX1 (cioè 50) inclusi.
Il passo successivo è ora la comprensione delle espressioni algebriche; come in BASIC, e in molti altri linguaggi ad alto livello, queste sono essenziali per scrivere programmi.
Supponiamo ora che A e B siano variabili e C sia un array; allora le seguenti sono espressioni lecite:

```
A
A+B
A★(A+B)
C(A+B★B★(A+B))
C(C(A+C(B))/B)
```

Queste invece non lo sono:

```
A++B
A+C (manca l'indice di C)
```

In generale sono lecite tutte le espressioni algebriche classiche; inoltre 1 e 0 sono interpretati, come in BASIC, come i valori logici VERO e FALSO; sono quindi lecite le seguenti espressioni:

```
A+(A>B)
A+(A>B OR C(A+B))
```

Attenzione, l'espressione

```
A> -1
```

non è lecita, in quanto '>' viene considerato un operatore in tutto e per tutto simile a '+' e generalmente non è accettata l'espressione:

```
A+ -1
```

Questa limitazione è comunque facilmente superabile (basta scrivere A>(-1)) e di scarsa importanza in confronto al vantaggio di poter mischiare a piacere operatori aritmetici e logici. Ecco una lista degli operatori in ordine di precedenza (ricordate la convenzione 1=vero, 0=falso):

- ODD (vale 1 se l'operando è dispari; 0 se è pari; esempio: ODD A, ODD(A+B★C(1)));
- NOT (vale 1 se l'operando è nullo, 0 altrimenti; esempio: NOT(A<B) è uguale a A<=B);
- ABS (vale il valore assoluto dell'operando, esempio: ABS A, ABS(A+B★C(A)));
- / (è l'operatore di divisione, come nell'Integer BASIC; bisogna però fare attenzione, in quanto in PL/Bit il risultato è approssimato ad un intero: 2/3 vale 0, non 0.666666 come in Applesoft);
- MOD (è l'operatore modulo, funziona esattamente come in Integer BASIC, non è il resto come inteso in algebra e bisogna quindi fare attenzione con i numeri negativi);
- ★, -, + (- è sia operatore unario come in -A, sia binario, come in A-B)
<, <=, =, <>, >=, > (sono i soliti operatori di confronto tra numeri; il risultato è 0 o 1 con le solite convenzioni: 0=falso, 1=vero);
- AND, OR (sono gli operatori logici, accettano qualsiasi valore, anche diverso da 0 e 1, con la convenzione che qualunque numero tranne 0 vale vero; il risultato è comunque uno 0 o un 1, esempio: (A OR B) AND A);

Possiamo ora finalmente introdurre il primo statement: l'assegnamento.
In BASIC si usa scrivere:

```
A=B
```

in PL/Bit c'è una leggera modifica, si scrive:

```
A:=B
```

L'uso del segno ':=' invece del vecchio '=' ha un significato ben preciso: quando in BASIC scriviamo A=B non intendiamo fare un'affermazione del tipo "il valore di A è pari al valore di B", ma intendiamo richiedere al calcolatore di assegnare alla variabile A il valore della variabile B; in

APPLE

Listato 2 - Programma Hilbert. Questo listato, tratto con leggere modifiche dal libro di Wirth "Algorithms+Data Structures = Programs", mostra come realizzare in PL/Bit schemi complessi di ricorsione.

effetti, questa differenza era riconosciuta nei primi dialetti del BASIC, che richiedevano si scrivesse: "LET A=B", poi la cosa è diventata facoltativa ed è andata rapidamente in disuso. Il segno ':=' è stato introdotto (credo per la prima volta) nel BNF, un formalismo per descrivere i linguaggi, ed è stato prontamente utilizzato nei linguaggi più recenti (Pascal ed altri), tanto che, dovendo il BNF trattare anche questi linguaggi, per evitare confusioni si usa ora in BNF il segno '::='; è importante comunque capire la differenza che passa tra il segno ':=' che indica assegnamento e l'operatore '=' che indica un confronto tra valori.
Naturalmente è possibile assegnare valori anche a elementi di un array, per esempio:

C(A+B★(A+1)):=A=B

(da notare qui la differenza tra il segno ':=' e il segno '=').
Passiamo ora a quegli statement che consentono al calcolatore di prendere delle decisioni e permettono quindi di scrivere dei programmi utili. Lo statement più semplice di questo tipo è:

IF <condizione> THEN <statement> ;

Si tratta in sostanza dello stesso statement del BASIC, con la solita convenzione che 0 significa falso, mentre qualunque altro numero significa

```
INTERFACE
 HPLOT 2,
 HPLOTTO 2,
 HCOLOR 1,
 USEPAGE 1,
 SHOWPAGE 1,
 FILLSCREEN 1;

CONST
 N=7,HO=128;

VAR
 I,H,X,Y,XO,YO,IT;

PROCEDURE A I;

 PROCEDURE B I;

  PROCEDURE C I;

   PROCEDURE D I;

   BEGIN IF I>O THEN
    BEGIN
     CALL A I-1;Y:=Y-H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
     CALL D I-1;X:=X-H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
     CALL D I-1;Y:=Y+H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
     CALL C I-1;
    END
   END;

  BEGIN IF I>O THEN
   BEGIN
    CALL B I-1;X:=X+H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
    CALL C I-1;Y:=Y+H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
    CALL C I-1;X:=X-H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
    CALL D I-1;
   END
  END;

 BEGIN IF I>O THEN
  BEGIN
   CALL C I-1;Y:=Y+H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
   CALL B I-1;X:=X+H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
   CALL B I-1;Y:=Y-H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
   CALL A I-1;
  END
 END;

BEGIN IF I>O THEN
 BEGIN
```

```
    CALL D I-1;X:=X-H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
    CALL A I-1;Y:=Y-H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
    CALL A I-1;X:=X+H;CALL HPLOTTO (X*3)/2,(Y*3)/2;
    CALL B I-1;
   END
  END;

  BEGIN IT:=0;
  REPEAT IT:=1-IT;
   CALL SHOWPAGE 1;CALL USEPAGE 1;
   CALL FILLSCREEN 0;CALL HCOLOR 3;
   I:=0;H:=H0;X0:=H/2;Y0:=X0;
   REPEAT
    IF IT=0 THEN CALL FILLSCREEN 0;
    I:=I+1;H:=H/2;
    X0:=X0+H/2;Y0:=Y0+H/2;
    X:=X0+28;Y:=Y0;CALL HPLOT (X*3)/2,(Y*3)/2;
    CALL A I;
   UNTIL I=N
  UNTIL 1=2
  END.
```

Seguito listato 1.

vero; lo statement può essere qualunque statement compreso l'assegnamento e lo stesso statement IF...; consideriamo i seguenti esempi:

IF A>B THEN B:=B+1;
IF A THEN A:=B;
IF A=B THEN IF A=C(0) THEN B:=5;
IF A=B AND A=C(0) THEN B:=5;

Da notare che i due ultimi statement fanno esattamente la stessa cosa.
Esiste una versione dello statement IF di cui in Applesoft si sente spesso la mancanza:

IF <condizione> THEN <statement> ELSE <statement>;

In questa versione dello statement IF lo statement dopo THEN viene eseguito se la condizione è vera, quello dopo ELSE se la condizione è falsa; consideriamo i seguenti esempi:

IF A=B THEN C(A):=B ELSE C(B):=A;
IF A<B THEN IF A=0 THEN B:=1 ELSE B:=A;

In quest'ultimo esempio si ha una certa ambiguità: ELSE è preceduto da due IF e viene spontaneo chiedersi a quale si riferisce, cioè se lo statement B:=A viene eseguito se non è A<B o se viene eseguito quando è A<B ma non A=0; cercherò di chiarire il problema con l'uso di alcune parentesi (che, sia chiaro, hanno solo una funzione grafica e non rientrano nella sintassi del linguaggio); nel primo caso avremmo:

IF A<B THEN (IF A = 0 THEN B:=1)
        ELSE B:=A;
nel secondo invece:

IF A<B THEN (IF A=0 THEN B:=1
                    ELSE B:=A);

È convenzione che l'interpretazione corretta sia la seconda, ma come ottenere, se necessario, che il calcolatore utilizzi la prima di queste due interpretazioni?
Questo ci porta direttamente al prossimo argomento: lo statement compound.
Ci si trova spesso, nei linguaggi strutturati, a dover mettere qualcosa di più complesso di un semplice statemente in un posto dove ci può stare un solo statement. Consideriamo la frase: 'se A<B allora poniamo A e B uguali a zero'.
È ovvio che questo non equivale a scrivere:

IF A<B THEN A:=0;B:=0;

perchè in questo caso B sarà posto uguale a zero anche se non è A<B. Si è pensato quindi a qualcosa di simile alle parentesi che ho usato prima: le parole BEGIN e END (ove la prima fa la funzione di parentesi aperta e la seconda quella di parentesi chiusa; riporto per dovere di cronaca che esiste un linguaggio, il C, che usa invece le parentesi graffe per questo scopo).
Per interpretare la frase precedente basta quindi scrivere:

IF A<B THEN
 BEGIN A:=0;B:=0 END;

E per ottenere la prima interpretazione dello statement scritto sopra scriveremo:

IF A<B THEN
 BEGIN IF A=0 THEN B:=1 END
ELSE B:=A;

APPLE

Figura 1 - I diagrammi sintattici illustrano in modo formale la sintassi di PL/Bit e costituiscono un utile mezzo per comprendere il motivo dei messaggi d'errore del compilatore. Il carattere @ è un comando proprio del compilatore, non una parola del linguaggio, e non è quindi riportato nei diagrammi.

Un'ultima cosa degna di nota sullo statement IF: il punto e virgola non va messo prima di ELSE:

```
IF A<B THEN A:=1;ELSE A:=0;
```

è sbagliato in quanto, come il Pascal, il PL/Bit usa il punto e virgola per separare gli statement e l'ELSE non è un nuovo statement, ma parte integrante dello statemet IF; sembrerà strano, ma, insieme al punto e virgola omesso tra gli statement e al segno '=' usato al posto di ':=', questo errore gareggia per i primi posti fra quelli più diffusi in Pascal.
Nel PL/Bit manca il GOTO tipico del BASIC; infatti, anche se l'uso di questo statement non implica di per sé che il programma risulterà illeggibile, si è notato che spesso il GOTO è usato male e si è quindi sostituito questo statement con altri due: il REPEAT...UNTIL... e il WHILE...DO...; potrà sembrare strano, ad un affezionato del BASIC, ma è stato dimostrato che tutto ciò che si riesce a fare con i calcolatori può essere descritto in termini di scelte (IF...THEN... e IF...THEN...ELSE...), sequenze (BEGIN...END) e iterazioni (WHILE...DO... e REPEAT...UNTIL...), e risulta in più, per esperienza pratica, che esprimendo i programmi in questo modo si ha qualcosa di facilmente leggibile e verificabile.
La sintassi per lo statement REPEAT...UNTIL... è la seguente:

REPEAT <statement>;<statement>... UNTIL <condizione>

il calcolatore eseguirà gli statement indicati, dopodiché valuterà la condizione: se questa risulterà falsa riprenderà ad eseguire gli statement, altrimenti proseguirà l'esecuzione del programma; consideriamo per chiarezza questi due programmi equivalenti:

```
I:=0;              10 I=0
REPEAT             20 REM CICLO
 A:=2★A;           30 A=2★A
 I:=I+1            40 I=I+1
UNTIL A>=B;        50 IF A<B THEN GOTO 20
```

Penso che la maggior leggibilità della prima forma sia evidente.
Molto simile a questo statement è il costrutto WHILE...DO...; in effetti la presenza di entrambi è un lusso che risulta però spesso comodo. La forma dello statement WHILE...DO.... è la seguente:

WHILE <condizione> DO <statement>

se la condizione è vera, lo statement viene eseguito e si torna a valutare la condizione, se no, il calcolatore salta lo statement e continua. Consideriamo, per esempio, questa coppia di programmi:

```
I:=0;              10 I=0
WHILE B<A DO       20 IF B>=A THEN GOTO 60
  BEGIN
    A:=2★A;        30 A=2★A
    I:=I+1         40 I=I+1
  END;             50 GOTO 20
                   60 ......
```

Il vantaggio del costrutto WHILE rispetto al REPEAT sta nel fatto che col primo, se la condizione è falsa in partenza il ciclo non viene eseguito, mentre col REPEAT il ciclo viene comunque eseguito almeno una volta.
Abbiamo definito virtualmente tutto quello che serve per scrivere programmi, tuttavia questi tenderebbero ad essere incredibilmente prolissi, se non ci fosse possibilità di definire dei sottoprogrammi più o meno equivalenti alle subroutine in BASIC.
Questa parte del linguaggio è un po' più complicata di quanto non lo sia in BASIC, ma è anche incredibimente più versatile, permettendo di scrivere con eccezionale facilità programmi ricorsivi (si vedano per esempio i programmi HILBERT e HANOI).
Una subroutine è dichiarata in PL/Bit nel seguente modo:

PROCEDURE <nome> <var1>,<var2>....;

ove <var1>,<var2>... sono variabili che possono essere in numero qualunque, oppure essere assenti.
Seguono le dichiarazioni (eventuali) di costanti, variabili ed array locali; può essere inoltre presente una dichiarazione di un'altra subroutine (parlerò di questo in seguito) ed infine uno statement

(che può essere anche del tipo BEGIN...END).
Una tipica subroutine, può essere la seguente:

```
PROCEDURE DELAY N;

CONST SCALEFACT=5;

VAR I;

BEGIN
 I:=5★N;
 WHILE I>0 DO I:=I-1
END;
```

Sono noti vari schemi con cui la variabile N, dichiarata nella lista dei parametri della subroutine, può essere legata al programma chiamante. Quella scelta è la seguente: il programma chiamerà la procedura DELAY con uno statement del tipo:

```
CALL DELAY <espressione>;
```

il valore dell'espressione verrà passato alla subroutine, che lo assegnerà alla sua variabile interna N, assolutamente invisibile dal programma chiamante; ciò significa che se all'interno della subroutine il valore di N viene modificato, questo riguarda soltanto la subroutine ed il programma chiamante non ha assolutamente modo di accorgersene.
Quindi, per esempio, dato il programma:

```
VAR A;

PROCEDURE ALFA A;

A:=5;


BEGIN
 A:=8;CALL ALFA A
END.
```

la variabile A dichiarata nella lista dei parametri della procedura ALFA non ha niente a che spartire con la A dichiarata come variabile nel programma principale ed inoltre la variabile A (del programma principale) varrà ancora 8 all'uscita dal programma stesso. Detto questo possiamo riscrivere in modo più compatto la procedura DELAY:

```
PROCEDURE DELAY N;

CONST SCALEFACT=5;

BEGIN
 N:=5★N;
 WHILE N>0 DO N:=N-1
END;
```

Naturalmente, il compilatore si aspetta che il numero di parametri formali dichiarati per una certa procedura corrisponda al numero di parametri con cui la procedura è chiamata; per esempio, data una procedura dichiarata come:
PROCEDURE ALFA A,B,C;...
se nel programma o in un'altra procedura si trova scritto:

```
CALL ALFA 2+3,3★6;
CALL ALFA 2,6,7;
CALL ALFA 3,6,7,8;
```

solo la seconda chiamata verrà considerata regolare.
Ma la caratteristica più interessante delle subroutine del PL/Bit è senza dubbio la possibilità che offrono di scrivere programmi ricorsivi; la ricorsione, a scapito di una certa difficoltà nella scrittura del compilatore, offre grandi vantaggi per il programmatore; nella prossima puntata parlerò del compilatore e dei problemi incontrati; per adesso consideriamo i vantaggi.
Guardiamo il programma Hanoi: potrà sembrare lungo, ma in realtà il 90% del codice serve solo per la grafica: il problema della torre di Hanoi è risolto dalla subroutine ricorsiva MOVE. Questa compattezza non è un merito di strane caratteristiche del linguaggio, ma esclusivamente della ricorsione; per convincercene consideriamo il seguente programma LISP (dialetto OWL computer):

```
(DE MOVE (ORG VIA DST NUM)
(COND
 (ZEROP NUM NIL)
 (T (MOVE ORG DST VIA (SUB1 NUM))
 (MOVEDISK ORG DST)
 (MOVE VIA ORG DST (SUB1 NUM)))))
```

Nulla hanno in comune il LISP e il PL/Bit, eccetto la ricorsione, ma la semplicità ottenuta è la stessa. Non che la ricorsione sia una panacea universale, ma esistono problemi come la torre di Hanoi o il disegno della curva di Hilbert per cui la soluzione ricorsiva appare in tutta la sua eleganza. Ma in cosa sussiste il supporto offerto alle subroutine ricorsive dal PL/Bit? In fondo, anche in BASIC possiamo scrivere una subroutine che richiama sé stessa:

```
1000 REM SUB HANOI
1100 GOSUB 1000
```

Già, ma il BASIC non genera variabili locali; ogni volta che in LISP o in PL/Bit una subroutine viene chiamata, eventualmente anche da se stessa, un nuovo spazio viene generato per le variabili locali, in modo che queste non interferiscano in alcun modo con le variabili della subroutine chiamante: quando la subroutine MOVE chiama sé stessa assegna alla variabile NUM della subroutine chiamata il valore NUM-1, dove però NUM è la variabile della subroutine chiamante. E' lo stesso discorso, in fondo, di quando si risolve ricorsivamente la torre di Hanoi in BASIC: si definisce un array, diciamo NUM(10), si tiene in qualche modo aggiornato un indice del livello di subroutine, diciamo DEPTH, e ogni volta che si ha bisogno del valore NUM, per una data profondità della subroutine, ci si riferisce a NUM(DEPTH). (In realtà si scopre che NUM è banalmente legato a DEPTH, cioè NUM=<dischi da spostare all'inizio>-DEPTH, però non esiste niente di così semplice per ORG, VIA e DST).
Ma, come in tutte le cose, una volta che ci si

prende gusto si vuole andare più in là: e se io volessi quattro subroutine A, B, C, D che si richiamano a vicenda? Beh, nessun problema, verrebbe da rispondere, e invece il problema c'è: anche se non si vede guardando i diagrammi sintattici, ogni cosa (procedura, costante, variabile, array) deve essere dichiarata prima dell'uso, quindi se A, B, C, D sono dichiarate nell'ordine, A non potrà mai chiamare D. La soluzione è semplice (anche se dubito che abbia nulla a che vedere con quello che Wirth intende con procedure interne ad altre procedure, comunque il metodo è molto comodo e ho deciso di seguirlo); basta dichiarare le procedure una dentro l'altra:

```
PROCEDURE A;
 PROCEDURE B;
  PROCEDURE C;
   PROCEDURE D;
   <codice per D>
  <codice per C>
 <codice per B>
<codice per A>
<codice programma principale>
```

Con questo schema ogni procedura è dichiarata prima dell'uso (per essere ortodossi una procedura A dichiarata dentro una B dovrebbe essere materiale riservato di questa e non essere visibile da un'altra procedura che non sia anch'essa interna a B, ma amo troppo le comodità offerte dalla ricorsione per essere ortodosso).
Qualcuno si sarà già accorto che manca qualcosa: possiamo fare cose molto belle e divertenti, ma un programma è interessante se dialoga col mondo esterno; se fa il saggio della montagna e non mostra niente non interessa a nessuno.
Il linguaggio offre due soluzioni al probema.
La prima è la dichiarazione di subroutine esterna, che deve comparire all'inizio del programma nella forma:

```
INTERFACE <name1> <num1>,<name2>
<num2>...;
```

Per esempio, data una dichiarazione:

```
INTERFACE HPLOT 2,HCOLOR 1;
```

il programma riconoscerà HPLOT come una subroutine con due parametri e HCOLOR come una subroutine con un parametro e accetterà quindi gli statement:

```
CALL HPLOT X+3★A,Y★5;
CALL HCOLOR 2★I;
```

Una completa trattazione di questo argomento è al di là degli scopi di questa puntata introduttiva; posso comunque anticipare che è presente nella libreria del linguaggio una nutrita serie di comandi grafici, che aspetta solo di essere dichiarata nella sezione INTERFACE di qualche programma. Un altro modo di accedere al linguaggio macchina, e quindi attraverso di esso a tutto il sistema, è tramite il comando α, che permette di inserire delle linee in assembler all'interno del codice prodotto dal compilatore; un esempio interessante è contenuto nel programma Hanoi, ove un loop in assembler è stato scritto per leggere un carattere da tastiera. Una completa comprensione della dichiarazione INTERFACE e del comando α richiede comunque una certa conoscenza del compilatore, che descriverò nella prossima puntata.

(Continua)

Sul listato del programma "Monitor per C 64", pubblicato sul n. 51, constatiamo un errore (molto banale) nelle linee 15320 e 15340, dove compare:

```
15320 ...          E$=MID$=(X$,6,4):...
15340 ...          E$=MID$=(X$,7,4):...
```

che vanno evidentemente sostituite, rispettivamente, con:

```
15320 ...          E$=MID$(X$,6,4):...
15340 ...          E$=MID$(X$,7,4):...
```

consigliamo, poi, di sostituire la linea 10910 con la seguente:

```
10910 IF NI>20 AND PR=0 THEN GOSUB14000
```

Con queste correzioni il programma girerà perfettamente, sempre che i DATA siano copiati con certosina attenzione.

ZX SPECTRUM

Dizionario

Sarà capitato a tutti coloro che hanno, o hanno avuto, la sfortuna di andare a scuola di dover fare una traduzione da o in una lingua straniera; esiste poi una categoria di gente continuamente alle prese con traduzioni a causa del loro lavoro.
Dal giorno in cui crollò la Torre di Babele ad oggi, è stata fatta poca strada verso l'unificazione del linguaggio né, d'altra parte, ne è stata fatta molta di più nel campo della traduzione automatica, anche se qualcosa si muove in questa branca dell'intelligenza artificiale.
A tutt'oggi, dunque, l'unico metodo per eseguire una traduzione è armarsi di pazienza e di un buon vocabolario e accingersi a tradurre frase per frase.

di **Luca Brigatti**

La parte senz'altro più noiosa, e per la quale si perde più tempo, in una traduzione è la ricerca, sul vocabolario, dei termini che non si conoscono, ma decisamente più seccante è il ricercare parole già cercate precedentemente, magari pochi minuti prima nell'ambito della stessa traduzione e già trovate, ma di cui proprio non si riesce a ricordare il significato.
Quando questo succede non resta che accingersi a ricercare il vocabolo, maledicendo l'inutile perdita di tempo, e a ritrovare il significato che, peraltro, avevamo "sulla punta della lingua".
Il programma proposto, fra l'altro, serve appunto a evitare questa seccatura.
Basterà, ogni qualvolta si stia facendo una traduzione, lasciare acceso il calcolatore con il programma inserito; quando si incontra una parola sconosciuta la si cerca sul vocabolario per la prima (che sarà anche l'ultima) volta, quindi si procede a memorizzare nel calcolatore parola e significato. Finita la traduzione procederemo a memorizzare su nastro i vocaboli.

Il risparmio di tempo e di fatica è evidente; l'incombenza di dover inserire i vocaboli in memoria, la prima volta, è abbondantemente compensata dalla rapidità con cui viene ritrovato tale vocabolo le volte successive.
Senza contare poi che i vocaboli in gioco sono necessariamente limitati e piuttosto ricorrenti, per cui la fase "inserimento", dopo diverse traduzioni, non verrà quasi più utilizzata, mentre la fase "ricerca" sarà sempre utile, traduzione dopo traduzione.

Il numero di vocaboli che può essere memorizzato è di circa 1200 x 2 parole di 15 lettere ciascuna in uno Spectrum con 48 Kbyte di RAM e di circa 180 x 2 (sempre di 15 lettere ciascuna) in uno Spectrum con 16 Kbyte di RAM.
È ovvio che, diminuendo il numero di lettere massimo assegnato ad ogni parola aumenteranno le parole memorizzabili.
Ad esempio, riducendo a 10 la lunghezza massima delle parole il numero di vocaboli memorizzabili sale a circa 280 x 2 in uno Spectrum a 16 Kbyte e a circa 1900 x 2 su un 48 Kbyte: va notato però che tale soluzione non è sempre consigliabile perché, anche con lingue come l'inglese, se è vero che la maggior parte dei vocaboli ha una lunghezza minore di 10 lettere, non sono comunque poche quelle che eccedono tale lunghezza, mentre sono molto meno quelle che eccedono le quindici lettere, e ciò è anche più vero con lingue come francese, tedesco o italiano.
Dopo aver digitato il programma (nel quale, per uno Spectrum 16 K, la linea 1240 andrà sostituita come segue:

```
1240 FOR w = VAL a$ (1, 1) TO 180)
```

questo dovrà essere salvato con l'istruzione:

```
SAVE "DIZIONARIO" LINE 1000
```

e, la prima volta (e solo la prima volta) che viene usato il programma si digitino le seguenti istruzioni:

```
DIM a$ (1200, 2, 15)
```

per uno Spectrum 48 Kbyte, oppure:

```
DIM a$ (180, 2, 15)
```

per uno Spectrum 16 Kbyte e quindi, per entrambi:

```
LET a$ (1, 1) = "2"
```

La variabile a$ (1, 1) funziona da contatore e contiene l'indice della variabile che deve essere inserita: ad esempio, se in memoria ci sono già 5

vocaboli, a$ (1, 1) sarà uguale a "6". All'inizio la prima variabile da inserire sarà con indice 2, ecco il perché dell'assegnazione vista più sopra.

Il programma stesso aggiornerà il contatore ad ogni nuovo inserimento.

A questo scopo si è scelta proprio una variabile stringa dimensionata, che altrimenti conterrebbe un vocabolo (quindi i vocaboli effettivamente memorizzabili, stante la DIM vista più sopra, sono in realtà 1199), perché tale variabile viene salvata e caricata con i vocaboli stessi, informando il calcolatore su quanti vocaboli sono già presenti, informazione che altrimenti il calcolatore non potrebbe ottenere se si usasse una variabile convenzionale, che perderebbe il suo valore ogni qualvolta il calcolatore viene spento.

Dopo aver per la prima volta usato il programma occorrerà salvare i vocaboli in un'area del nastro succesiva a quella in cui è stato salvato il programma, avendo cura di annotare il valore del counter, poiché tutte le successive memorizzazioni avverranno in quella stessa area, sovrapponendo i nuovi file ai vecchi.

Il programma stesso, una volta caricato, cariche-

```
   0 >REM        DIZIONARIO
         ITALIANO-INGLESE
         INGLESE-ITALIANO
         By [illegible] Luca 1984

1010 LOAD "termini" DATA a$()
1020 GO SUB 1510: PAPER 7: INK 0
 : BORDER 7: CLS
1030 CLS : PRINT "Premi:"''
1040 PRINT " 0 Per ricercare"'"
1 Per inserire"'" 2 Per salvare
i nuovi files"'" 3 Per listare i
 nomi"'" 4 Per correggere"
1050 POKE 23658,8: REM [illegible]
1060 PAUSE 0: LET a=VAL INKEY$
1070 CLS
1080 GO TO 1090*(a=0)+1230*(a=1)
+1330*(a=2)+1390*(a=3)+1460*(a=4
)
1090 REM RICERCA
1100 PRINT "Disponi della parola
..."''"1) Italiana"'"2) Inglese"
1110 PAUSE 0: LET li=VAL INKEY$:
 CLS
1120 INPUT "Scrivi la parola ( S
TOP per   uscire)"'x$
1130 IF x$=" STOP " THEN GO TO 1
030
1140 LET lu=LEN x$
1150 FOR r=2 TO VAL a$(1,1)-1
1160 IF a$(r,li, TO lu)=x$ THEN
GO TO 1190
1170 NEXT r
1180 GO TO 1120
1190 PRINT a$(r,1);a$(r,2)
1200 INPUT "Desideri continuare
la ricerca? [S/N] ";c$
1210 IF c$="S" THEN GO TO 1170
1220 GO TO 1120
1230 REM INSERIMENTO
1240 FOR w=VAL a$(1.1) TO 1200
1250 PRINT w'
1260 INPUT "Scrivi la parola ING
LESE ( STOP per uscire)"'b$
1270 IF b$=" STOP " THEN LET a$(
1,1)=STR$ w: GO TO 1030
1280 LET a$(w.2)=b$
1290 INPUT "Scrivi la parola ITA
LIANA"'a$(w.1)
1300 CLS : NEXT w
1310 PRINT FLASH 1;"NON C'E' PIU
' SPAZIO IN MEMORIA PER ALTRI TE
RMINI"
1320 PAUSE 250: GO TO 1030
1330 REM SALVATAGGIO SU NASTRO
1340 SAVE "termini" DATA a$()
1350 PRINT "Riposiziona il nastr
o,inserisci il cavetto EAR per l
```

Figura 1 - Listato del programma "Dizionario".

ZX SPECTRUM

Seguito figura 1.

```
a verifica, quindi fa partire il
 nastro e  premi qualsiasi tast
o."'"(Premi ""N"" se non vuoi la
 verifica.)"
1360 PAUSE 0: IF INKEY$="N" THEN
 GO TO 1030
1370 VERIFY "termini" DATA a$()
1380 GO TO 1030
1390 REM LISTA DI NOMI
1400 FOR l=2 TO VAL a$(1,1)-1
1410 PRINT l;")  ";a$(l,1);a$(l,2
)
1420 PRINT
1400 NEXT l
1440 INPUT "Premi ""ENTER"" per
tornare al Menu'.";z$
1450 GO TO 1030
1460 REM CORREZIONE
1470 INPUT "Scrivi il numero del
 termine da correggere ";nc
1480 INPUT "Scrivi la parola ING
LESE "'a$(nc,2)
1490 INPUT "Scrivi la parola ITA
LIANA "'a$(nc,1)
1500 GO TO 1030
1510 REM  PRESENTAZIONE
1520 BORDER 2: PAPER 6: CLS
1530 BRIGHT 1: OVER 1
1540 RESTORE 1710
1550 FOR y=1 TO 3
1560 READ z$,pfy,ri
1570 LET in=(32-LEN z$)/2-1
1580 FOR z=1 TO LEN z$
1590 LET i=INT (RND*5)
1600 LET ax=INT (RND*2)*230: LET
 px=INT (RND*25)+ax
1610 REM  px 230 o px 25
1620 LET ay=INT (RND*2)*150: LET
 py=INT (RND*25)+ay
1630 REM  py 150 o py 25
1640 LET pfx=(z+in)*8+4
1650 LET ix=pfx-px: LET iy=pfy-p
y
1660 INK i
1670 PLOT px,py: DRAW ix,iy
1680 BEEP .125,INT (RND*60)-20
1690 PLOT px,py: DRAW ix,iy
1700 BEEP .125,INT (RND*60)-20
1710 PRINT FLASH 1;AT ri,in+z;z$
(z)
1720 NEXT z: NEXT y
1730 PRINT INK 9;AT 21,6;"By Bri
gatti Luca 1983"
1740 BRIGHT 0: OVER 0
1750 PAUSE 300: RETURN
1760 DATA "DIZIONARIO",108,8,"IT
ALIANO-INGLESE",92,10," ENGLISH-
ITALIAN",76,12
```

rà i vocaboli già memorizzati. Verrà quindi chiamata la routine alla linea 1460, routine che contiene la presentazione.
Terminata la presentazione apparirà il menu che comprende cinque opzioni:
0 ricerca;
1 inserimento;
2 salvataggio dei nuovi file;
3 lista dei nomi in memoria;
4 correzione.
Quindi viene attesa una scelta alla linea 1060.
La POKE alla linea 1050 serve per abilitare automaticamente i caratteri maiuscoli (caps lock).

Una volta fatta la scelta la linea 1080 provvede a far proseguire il programma dal punto in cui viene effettuata la funzione selezionata dal menu; tale linea è una simulazione della istruzione: ON... GO TO... che manca sullo Spectrum: ammettendo di aver scelto l'opzione 3 (lista dei nomi) solo la condizione logica a = 3 sarà vera (valore 1) e le altre saranno false (valore 0), per cui il risultato dell'espressione che è l'argomento del GO TO sarà:
$1090 \times 0 + 1230 \times 0 + 1330 \times 0 + 1390 \times 1 + 1460 \times 0 = 1390$
ed è proprio alla linea 1390 che inizia la routine

che lista le parole memorizzate.

Le linee da 1090 a 1230 contengono la routine "ricerca".

Come si è detto, per effettuare una ricerca basta inserire la parola da ricercare: il programma visualizza immediatamente tale parola, insieme alla traduzione.

Una caratteristica che rende molto versatile e potente questo programma risiede nel fatto che la parola di n lettere da ricercare viene confrontata con la parte iniziale lunga n lettere di ogni stringa, ignorando tutte le altre lettere (o spazi) che possono seguire.

Ad esempio, digitando BOOK (libro) verrà visualizzato anche BOOKMAKER (allibratore), sempre che anch'esso sia presente in memoria, oppure digitando H verranno visualizzate tutte le parole che iniziano per H e così via.

Questo ci consente, al momento dell'inserimento, di riempire gli spazi che avanzano come più ci fa comodo, senza che questo interferisca con la ricerca; ad esempio potremmo scrivere:

GO WENT GONE
RUN RUN RUN
MOUSE pl. MICE
MAN pl. MEN
OX pl. OXEN
CHEQUE am CHECK
COLOUR am COLOR
HEART a. fig.
ecc...

Dove, ovviamente, a. fig. sta per "anche figurato", pl. sta per "plurale", am per "americano" e dove WENT e GONE e anche RUN e RUN sono i passati e participi passati rispettivamente di "to go" e di "to run".

Queste stringhe verranno visualizzate per intero, insieme alla loro traduzione, digitando: GO; RUN; MOUSE; MAN; OX; CHEQUE; COLOUR e HEART come pure: ANDARE; CORRERE; TOPO; UOMO; BUE; ASSEGNO; COLORE e CUORE.

Tornando a ciò che si è detto prima, se è vero che stringhe di quindici caratteri consentono meno parole che non stringhe di lunghezze minori, è anche vero che maggiore è la lunghezza di una stringa e maggiori sono le informazioni che possono essere memorizzate relativamente a ogni vocabolo.

La ricerca è di tipo sequenziale e non dicotomico, come parrebbe più conveniente avendo a che fare con liste ordinabili, poiché tale ricerca può essere effettuata sia sulla lista italiana sia sulla lista inglese, mentre l'ordinamento alfabetico può essere fatto solo su una lista; se venisse fatto su entrambe, non vi sarebbe più corrispondenza nel contenuto dell'array a$; cioè, se per esempio a$ (5, 1) contiene "UOMO", a$ (5, 2) deve contenere "MAN".

Il programma ci chiede quindi se vogliamo continuare la ricerca (nel caso che la parola che cerchiamo abbia più di un significato).

Arrivati a fondo lista, rispondendo "N" a tale richiesta; viene sottoposto un altro vocabolo da cercare; rispondendo STOP (Symbol shift + A) si torna al menu.

Le linee da 1230 a 1320 contengono la routine inserimento, il cui uso è pilotato dal programma stesso.

Si rimane in questa routine finché non si preme STOP.

Le linee da 1330 a 1380 contengono la routine di salvataggio dei nomi su nastro: l'uso di tale routine è pilotato dal programma stesso; si abbia solo cura di posizionare correttamente il nastro.

Le linee da 1390 a 1450 contengono la routine che scrive sul video i vocaboli memorizzati.

Per avere la stampa su carta basta sostituire le PRINT alle linee 1410 e 1420 con LPRINT.

Le linee da 1460 a 1500 contengono la routine di correzione.

Dopo aver annotato il numero del vocabolo errato lo si inserirà all'INPUT di linea 1470, cui seguirà l'inserimento del vocabolo corretto.

Le linee da 1510 a 1760 contengono la presentazione.

Ritengo superfluo raccomandare di non usare, qualora il programma si dovesse fermare, le istruzioni RUN, CLEAR o NEW, ma di dare GOTO 1030 per tornare al menu principale.

Il programma, così com'è, funziona per le lingue italiana e inglese; con poche modifiche alle stringhe nelle linee 0, 1260, 1480 e 1760 (peraltro non necessarie al buon funzionamento del programma), e con l'aggiunta dei caratteri definibili dall'utente, funziona anche con altre lingue, come il francese (ç), il tedesco (ü, β), lo spagnolo (ñ), le lingue scandinave (ö, å, ø) e, con modifiche più sostanziali, anche con lingue che adottano un alfabeto diverso, come il greco, il russo, l'ebraico e altre, o addirittura con lingue come il cinese o il giapponese (l'autore ha già scritto le versioni per il greco ed il giapponese).

CONSIGLIO PER MEMORY ALPHA IV (Bit n. 48)

Introducendo nei campi più di 9 caratteri si hanno dei problemi. Per evitarli è sufficiente eliminare alla riga 40180 i caratteri di "reverse on" e "reverse off" immediatamente precedenti e seguenti lo Z$.

Calcolo di strutture

Con un programma adatto ad una tipologia di strutture assai ricorrenti si può ridurre di più dell'80% il lavoro di "input" dei dati e risparmiare sulla occupazione di memoria, in modo da rendere accessibile a piccoli computer la risoluzione di problemi di medie dimensioni.

di **Giovanni Malato**

Il calcolo strutturale, avvalendosi dell'uso del computer, utilizza moderni metodi generali di analisi, basati sull'algebra matriciale (vedi l'articolo "Le nuove frontiere del calcolo strutturale" di G. Forcolini su **Bit** dell'Aprile 1983). È in questo modo possibile studiare strutture di configurazione irregolare qualsiasi, sfruttando le proprietà della matrice di rigidezza della struttura con l'algoritmo di Cholesky.
Ma, ed è questo quello che si vuole sostenere in questa sede, il riferirsi sempre ed in ogni caso ad un programma generale non è sempre conveniente, per almeno due motivi:
1) il metodo generale deve naturalmente prevedere una casistica che, per casi ricorrenti e delimitati, può essere assai semplificata, con un risparmio sensibile della memoria del computer;
2) il metodo generale, pur ottenendo le soluzioni richieste, non può, per sua natura, essere approfondito e specificato con le caratteristiche che può invece fornire un programma più ristretto, ma più aderente alle necessità del progettista.

Un concreto paragone mostrerà in maniera più evidente quanto sopra enunciato; con riferimento all'esempio riportato più avanti notiamo: nel metodo della matrice di rigidezza ogni punto di applicazione del carico è individuato come un nodo, per cui per ogni asta, o tratto di asta, a seconda della "discretizzazione", è necessario fornire: le due coordinate iniziali, le due coordinate finali, l'area, il momento d'inerzia, le tre più tre componenti di spostamento del punto iniziale e del punto finale ed infine i carichi, cioè 13 dati per ognuna delle 13 travi, o tratti di trave, e per ognuno degli 11 pilastri dell'esempio. Ancora, riferendomi sempre all'esempio qui di seguito, per un telaio ad 11 nodi avremo 3 componenti di spostamento per ogni nodo, più 3 componenti per le cinque forze concentrate, in totale 48 incognite per una struttura che non può nemmeno essere considerata di media dimensione.
In conclusione, un grande lavoro di "input" (più di 300 dati), che può scoraggiare l'utente, ed un grande impegno di memoria e cioè necessità di elaboratori costosi non alla portata quindi di studenti e di piccoli liberi professionisti.

Se, viceversa, decidiamo di utilizzare un programma che sia limitato al solo caso, che però è assai ricorrente, delle strutture a maglie rettangolari regolari e con aste a momento di inerzia costante, ci accorgiamo che è assai elevata la semplificazione in fase di input giacché, dopo aver fornito, una volta per tutte, le luci delle campate e le altezze dei vari piani, dovremo solo fornire le rigidezze delle aste ed i carichi: in totale poco più di 40 dati per cui il rapporto è, in questo caso, di 1 a 7; l'algoritmo di soluzione si basa su una procedura di calcolo che si ripete per tutti gli 11 nodi, il che vuol dire basso dimensionamento dei vettori e delle matrici; la precisione dei risultati dipende solo dal numero di iterazioni che assolutamente non impegna la capacità di memoria del computer, il che vuol dire possibilità di utilizzare computer dal prezzo assai contenuto.

Ancora un aspetto assai importante da mettere in luce: con un programma adatto a questa tipologia di strutture si possono ottenere dati particolarmente utili, che difficilmente potrebbero essere raggiunti con un programma del tutto generale, che non può essere così duttile. Con il programma che presento ecco quali altri dati si possono avere: individuazione della ascissa, nonché valore del momento massimo positivo per un corretto dimensionamento delle travi in ogni campata; calcolo automatico della sommatoria degli sforzi assiali sui pilastri, piano per piano, con la possibilità quindi di dimensionamento completo degli stessi; input dei dati, senza necessità di una preparazione preliminare (cioè il computer calcola automaticamente le rigidezze ed i momenti di incastro perfetto a partire dai dati del progetto architettonico); facilità di controllo della validità dei dati in output (sommatoria nulla dei momenti intorno al nodo e degli sforzi taglianti nei pilastri, per ogni piano).

Analisi del programma

Il programma si divide in tre sezioni: dalla linea 100 alla linea 1390 sono contenute le istruzioni per l'input; dalla linea 1400 alla linea 1690 (con un GOSUB alla linea 2390 fino alla linea 2430) sono contenute le istruzioni per l'algoritmo risolutivo; dalla linea 1700 alla linea 2380 sono contenute le istruzioni di output.

Per i dimensionamenti delle righe 120 e 130 si è preferito utilizzare, fin dove possibile, i vettori che, a parità di elementi, occupano meno spazio delle matrici bidimensionali (che sarebbero state più comode); sempre nell'intento di risparmiare spazio in memoria le variabili RN, RO, RE, RS, che all'inizio sono i fattori di rotazione del metodo Kani e che non intervengono più da un certo punto in poi, sono state riutilizzate per il calcolo dei momenti di estremità definitivi.

Riprendiamo l'esame dall'inizio: la prima sezione contiene alcune opzioni alle linee 190, 260, 880: si chiede cioè all'utente di decidere se desidera che il calcolo venga eseguito tenendo conto o meno degli spostamenti orizzontali (190); se desidera introdurre direttamente le rigidezze delle aste o se desidera che il calcolo venga eseguito automaticamente dal computer a partire dalle dimensioni di progetto (260); il calcolo delle rigidezze, per sezioni rettangolari, viene eseguito in quest'ultimo caso alle righe 420 e 730. L'ultima opzione riguarda la decisione di immettere direttamente i momenti di incastro o far eseguire il calcolo dal computer (880); in quest'ultimo caso occorrerà fornire il carico uniforme su ogni trave (1070) e l'ascissa e l'intensità di una eventuale forza concentrata (1130); il calcolo di momento e taglio di incastro perfetto viene eseguito con le istruzioni da 1090 a 1200; nel caso l'utente desiderasse immettere i momenti di incastro perfetto direttamente, la istruzione 930 rimanda la prosecuzione del programma alla linea 1220.
Una caratteristica importante riguarda la previsione di sbalzi, per esempio balconi, a sinistra del primo nodo del piano o a destra dell'ultimo nodo del piano; le istruzioni relative sono alle righe da 950 a 1050.

I momenti di estremità definitivi sono ottenuti con le istruzioni dalla linea 1750 alla linea 1920. L'ascissa di momento massimo positivo si ottiene trovando il punto dove si annulla il taglio (linee da 2030 a 2180). Alla linea 2280 inizia il procedimento per il dimensionamento dei pilastri: partendo dall'alto si trova la sommatoria di tutti gli sforzi assiali dei pilastri, piano per piano, così da determinare a tutti i piani i carichi dei singoli pilastri.
Tutto il programma utilizza la RAM del BASIC non esteso del TI99/4A e cioè circa 14,5 Kbyte; con il dimensionamento proposto si possono studiare telai di 36 nodi con un massimo di 10 piani o di 6 campate, cioè strutture di medie dimensioni.
Per ulteriori dettagli o informazioni vi prego di scrivermi in Via Hajech 2 - 20129 Milano.

Bibliografia

W.M. Jenkins ed altri: BASIC Computing for civil engineers, 1983 Van Nostrand (UK)
M. Capurso: Introduzione al calcolo automatico delle strutture, 1983 E.S.A.C. Roma
G. Kani: Calcolo dei telai multipli, 1974 Pirola, Milano.

● IMMISSIONE DEI DATI:

NUMERO DELLE CAMPATE
3
NUMERO DEI PIANI
3

NODI FISSI 1
NODI SPOSTABILI 2
2

MOMENTI D'INERZIA 1
DIMENSIONI PILASTRI/TRAVI 2
1

● SCRIVI IL RAPPORTO K=J/L PER I PILASTRI CHE ARRIVANO AL PIANO N. 1
PILASTRO (1-1) **0.2** PILASTRO (1-2) **0.2**
PILASTRO (1-3) **0.3** PILASTRO (1-4) **0.3**

● SCRIVI IL RAPPORTO K=J/L PER I PILASTRI CHE ARRIVANO AL PIANO N. 2
PILASTRO (2-1) **0.2** PILASTRO (2-2) **0.2**
PILASTRO (2-3) **0.2** PILASTRO (2-4) **0.2**

● SCRIVI IL RAPPORTO K=J/L PER I PILASTRI CHE ARRIVANO AL PIANO N. 3
PILASTRO (3-1) **0.1** PILASTRO (3-2) **0.1**
PILASTRO (3-3) **0.1** PILASTRO (3-4) **0**

● SCRIVI IL RAPPORTO K=J/L PER LE TRAVI DEL PIANO N. 1

TRAVE 1 **1.5** TRAVE 2 **1.2** TRAVE 3 **1.2**
TRAVI DEL PIANO N. 2
TRAVE 1 **1.5** TRAVE 2 **1.2** TRAVE 3 **1.2**

TRAVI DEL PIANO N. 3
TRAVE 1 **1.5** TRAVE 2 **1.2** TRAVE 3 **0**

● VALORI RIGIDEZZE
CALCOLO MOMENTI E TAGLIO
2; ALTEZZA DEL PIANO N. 1
EVENTUALE MOMENTO A SINISTRA DEL NODO (1-1)
0
TAGLIO A SINISTRA DEL NODO (1-1)
0
SCRIVI LA LUCE DELLA CAMPATA N.1
4
● SCRIVI IL CARICO UNIFORME IN KG/M SULLA TRAVE N. 1 AL PIANO N. 1
1.8
● SCRIVI L'ASCISSA DA SINISTRA E L'INTENSITA' DELLA FORZA
0
0
● SCRIVI LA LUCE DELLA CAMPATA N. 2
6
● SCRIVI IL CARICO UNIFORME IN KG/M SULLA TRAVE N. 2 AL PIANO N. 1
1.8
● SCRIVI L'ASCISSA DA SINISTRA E L'INTENSITA' DELLA FORZA
4
7.2
● SCRIVI LA LUCE DELLA CAMPATA N. 3
6
● SCRIVI IL CARICO UNIFORME IN KG/M SULLA TRAVE N. 3 AL PIANO N. 1
1.8
● SCRIVI L'ASCISSA DA SINISTRA E L'INTENSITA' DELLA FORZA
4
7.2
EVENTUALE MOMENTO A DESTRA DEL NODO (1-4)
0
TAGLIO A DESTRA DEL NODO (1-4)
0
ALTEZZA DEL PIANO N. 2
4
EVENTUALE MOMENTO A SINISTRA DEL NODO (2-1)
0
TAGLIO A SINISTRA DEL NODO (2-1)
0
● SCRIVI IL CARICO UNIFORME IN KG/M SULLA TRAVE N. 1 AL PIANO N. 2
1.8

IL PROCEDIMENTO SI RIPETE PER TUTTI I PIANI.

Figura 1 - Un esempio di schema di telaio. In neretto la risposta dell'utente.

Figura 2 - Emissione dei dati.

MOMENTI ALL'INCASTRO DELLA FONDAZIONE:	VERIFICHE:		SOMMATORIA DEGLI SFORZI NEI PILASTRI:			
	SOMMATORIA DEI MOMENTI IN OGNI NODO = 0			AL PIANO N. 1	AL PIANO N. 2	AL PIANO N. 3
COLONNA N. 1 0.28	SOMMA DEGLI SFORZI TAGLIANTI IN OGNI PIANO = 0		COLONNA N. 1	1.37	3.87	6.37
COLONNA N. 2 0.44	SOMMA DEI CARICHI	SFORZI NORMALI PIL.	COLONNA N. 2	15.08	26.47	38.01
COLONNA N. 3 0.34	1.8 · (16 ·2+10) + = 75.6 +	6.37 + 38.01 +	COLONNA N. 3	8.75	29.75	50.31
COLONNA N. 4 −0.62	+ 7.2 · 5 = 111,6	50.31 + 16.91 = 111.6	COLONNA N. 4	0.0	8.32	16.91

	NODI	MOMENTI				
		OVEST	SUD	EST	NORD	
PIANO 1	(1 - 1)	0	0.3	−0.80	0.5	= 0
	(1 - 2)	5.17	0.61	−6.71	0.93	= 0
	(1 - 3)	12.43	0.29	−13.11	0.39	= 0
	(1 - 4)	3.42	−1.65	0	−1.78	= 0
PIANO 2	(2 - 1)	0	0.52	−0.64	0.12	= 0
	(2 - 2)	5.05	0.91	−6.5	0.53	= 0
	(2 - 3)	13.17	0.46	−13.22	−0.41	= 0
	(2 - 4)	1.94	−1.94	0	0	= 0
PIANO 3	(3 - 1)	0	0.063	−0.063	0	= 0
	(3 - 2)	8.98	0.69	−9.67	0	= 0
	(3 - 3)	1.00	−1.00	0	0	=0

						SFORZI TAGLIANTI PILASTRI				
		TAGLIO a Sx.	MOMENTO	POS. MAX	TAGLIO a Dx.	COL. N.1	COL. N.2	COL. N.3	COL. N.4	
PIANO 1	TRAVE N. 1	2.51	0.946	a metri 1.39 da S.	4.69	−0.146	−0.263	−0.158	0.567	= 0
	TRAVE N. 2	6.85	6.31	a metri 3.81	11.15					
	TRAVE N. 3	9.41	10.51	a metri 4.00	8.59					
PIANO 2	TRAVE N. 1	2.50	1.09	a metri 1.39	4.70	−0.255	−0.46	−0.212	0.928	= 0
	TRAVE N. 2	6.69	5.93	a metri 3.72	11.31					
	TRAVE N. 3	9.68	11.10	a metri 4.00	8.32					
PIANO 3	TRAVE N. 1	1.37	0.46	a metri 0.76	5.83	−0.053	−0.347	0.4	0.0	= 0
	TRAVE N. 2	9.25	12.91	a metri 4.00	8.75					

N.B.: CONFRONTA CON L'ESEMPIO CONTENUTO NEL LIBRO: "CALCOLO DEI TELAI MULTIPLI" DEL DR. ING. G. KANI

Seguito figura 2.

TI 99/4A

Listato 1 - Il programma che svolge il calcolo strutturale.

```
100 REM  GIOVANNI MALATO,VIA HAJ
ECH 2,20129 MILANO.TEL.02/741952

110 CALL CLEAR
120 DIM HP(11),KS(37),KE(37),MS(
37),MD(37),RN(37),RO(37),RE(37),
RS(37),KO(37),KN(37),LC(7),TD(37
),TS(37),QL(37)
130 DIM F(37),ASS(37),MPOS(37),E
I(7),ES(11),KD(11,7),MFON(7),N(1
1,7),QW(11,7)
140 PRINT "TELAI MULTIPLI"
150 PRINT "--------------------
-------"
160 INPUT "NUMERO DELLE CAMPATE
       ":AC
170 INPUT "NUMERO DEI PIANI
       ":BP
180 PRINT "--------------------
-------"
190 INPUT "NODI FISSI
       1
   NODI SPOSTABILI            2"
:Z
200 IF Z=1 THEN 220
210 IF Z<>2 THEN 190
220 PRINT "--------------------
-------"
230 M=0
240 FOR NP=1 TO BP
250 IF NP>1 THEN 270
260 INPUT "VALORI RIGIDEZZE
       1
   DIMENSIONI PILASTRI/TRAVI   2"
:L
270 IF L=1 THEN 290
280 IF L<>2 THEN 260
290 PRINT :::
300 PRINT "ALTEZZA DEL PIANO N."
;NP
310 INPUT HP(NP)
320 PRINT :::
330 IF L=1 THEN 360
340 PRINT "SCRIVI LARGHEZZA E BA
SE DEI PILASTRI CHE TERMINANO AL
   PIANO N.";NP
350 GOTO 370
360 PRINT "SCRIVI IL RAPPORTO K=
J/L PER I PILASTRI CHE ARRIVANO
AL PIANO N.";NP
370 FOR NC=1 TO AC+1
380 M=M+1
390 PRINT "PILASTRO (";NP;"-";NC
;")";
400 IF L=1 THEN 440
410 INPUT HF,BF
420 KS(M)=HF^3*BF/12/HP(NP)
430 GOTO 450
440 INPUT KS(M)
450 EDD(NP)=EDD(NP)+KS(M)
460 NEXT NC
470 INPUT "SE HAI SBAGLIATO PREM
I ""H""     ":RY$
480 IF RY$<>"H" THEN 520
490 M=M-(AC+1)
500 EDD(NP)=0
510 GOTO 250
520 NEXT NP
530 PRINT :::
540 M=0
550 PRINT "--------------------
-------"
560 FOR NP=1 TO BP
570 FOR NC=1 TO AC
580 IF NP>1 THEN 630
590 PRINT :::
600 PRINT "LUCE DELLA CAMPATA N.
";NC
610 INPUT LC(NC)
620 PRINT :::
630 IF L=1 THEN 670
640 IF NC>1 THEN 690
650 PRINT "SCRIVI L'ALTEZZA E LA
 BASE  DELLA TRAVE DEL PIANO N."
:NP
660 GOTO 690
670 IF NC>1 THEN 690
680 PRINT "SCRIVI IL RAPPORTO K=
J/L PER LE TRAVI DEL PIANO";NP
690 M=M+1
700 PRINT "TRAVE (";NC;"-";NC+1;
")";
710 IF L=1 THEN 750
720 INPUT HT,BT
730 KE(M)=HT^3*BT/12/LC(NC)
740 GOTO 760
750 INPUT KE(M)
760 NEXT NC
770 INPUT "SE HAI SBAGLIATO PREM
I ""H""    ":RS$
780 IF RS$<>"H" THEN 810
790 M=M-AC
800 GOTO 570
810 M=M+1
820 NEXT NP
830 M=0
840 PRINT :::
850 FOR NP=1 TO BP
860 IF NP>1 THEN 890
870 PRINT :::
880 INPUT "MOMENTI D'INCASTRO
       1
   CALCOLO MOMENTI E TAGLIO    2"
:Q
890 IF Q=1 THEN 910
900 IF Q<>2 THEN 880
910 FOR NC=1 TO AC+1
920 M=M+1
930 IF Q=1 THEN 1220
940 IF NC<>1 THEN 990
950 PRINT "EVENTUALE MOMENTO A S
INISTRA DEL NODO (";NP;"-";NC;")
"
960 INPUT MS(M)
970 PRINT "TAGLIO A SINISTRA DEL
       NODO (";NP;"-";NC;")"
980 INPUT TS(M)
990 IF NC=(AC+1)THEN 1020
```

Seguito listato 1.

```
1000 IF NP=1 THEN 1070
1010 IF NC<>(AC+1)THEN 1070
1020 PRINT "EVENTUALE MOMENTO A
DESTRA  DEL NODO (";NP;"-";NC;")
"
1030 INPUT MD(M)
1040 PRINT "TAGLIO A DESTRA DEL
         NODO (";NP;"-";NC;")"
1050 INPUT TD(M)
1060 GOTO 1210
1070 PRINT "SCRIVI IL CARICO UNI
FORME INKG/M SULLA TRAVE (";NC;"
-";NC+1;")";"AL PIANO N.";NP
1080 INPUT QL(M)
1090 MD(M)=-QL(M)*LC(NC)^2/12
1100 MS(M+1)=QL(M)*LC(NC)^2/12
1110 TD(M)=QL(M)*LC(NC)/2
1120 TS(M+1)=QL(M)*LC(NC)/2
1130 INPUT "SCRIVI L'ASCISSA DA
SINISTRA E L'INTENSITA' DELLA FO
RZA ? ":ASS(M)
1140 INPUT F(M)
1150 IF ASS(M)=0 THEN 1210
1160 Z1=LC(NC)-ASS(M)
1170 MD(M)=MD(M)+(-F(M)*ASS(M)*Z
1^2)/(LC(NC)^2)
1180 MS(M+1)=MS(M+1)+F(M)*ASS(M)
^2*Z1/(LC(NC)^2)
1190 TD(M)=TD(M)+F(M)*Z1/LC(NC)

1200 TS(M+1)=TS(M+1)+F(M)*ASS(M)
/LC(NC)
1210 GOTO 1240
1220 PRINT "SCRIVI I MOMENTI A S
INISTRA,DESTRA DEL NODO (";NP;"-
";NC;")"
1230 INPUT MS(M),MD(M)
1240 A=-2*(KE(M)+KS(M)+KE(M-1)+K
S(M+AC+1))
1250 RN(M)=KS(M+AC+1)/A
1260 IF NC=1 THEN 1290
1270 RO(M)=KE(M-1)/A
1280 IF NC=AC+1 THEN 1300
1290 RE(M)=KE(M)/A
1300 RS(M)=KS(M)/A
1310 KD(NP,NC)=-1.5*KS(M)/EDD(NP
)
1320 NEXT NC
1330 INPUT "SE HAI SBAGLIATO PRE
MI ""H""     ":RT$
1340 IF RT$<>"H" THEN 1370
1350 M=M-(AC+1)
1360 GOTO 860
1370 NEXT NP
1380 CALL CLEAR
1390 PRINT "ATTENDERE PREGO"
1400 PRINT :::
1410 GOSUB 2390
1420 FOR E=1 TO 10
1430 M=0
1440 FOR NE=1 TO BP
1450 ES(NE)=0
1460 FOR NQ=1 TO AC+1
1470 M=M+1
1480 IF NE=1 THEN 1510
1490 ES(NE)=ES(NE)+KS(M)+KN(M-(A
C+1))
1500 GOTO 1520
1510 ES(NE)=ES(NE)+KS(M)
1520 NEXT NQ
1530 FOR NR=1 TO AC+1
1540 QW(NE,NR)=ES(NE)*KD(NE,NR)

1550 NEXT NR
1560 NEXT NE
1570 M=0
1580 FOR NP=1 TO BP
1590 FOR NC=1 TO AC+1
1600 M=M+1
1610 IF E=1 THEN 1660
1620 IF M>(AC+1)THEN 1650
1630 ACC=MS(M)+MD(M)+KE(M-1)+KO(
M+1)+KS(M+AC+1)+QW(NP,NC)+QW(NP+
1,NC)
1640 GOTO 1660
1650 ACC=MS(M)+MD(M)+KE(M-1)+KO(
M+1)+KS(M+AC+1)+KN(M-(AC+1))+QW(
NP,NC)+QW(NP+1,NC)
1660 GOSUB 2390
1670 NEXT NC
1680 NEXT NP
1690 NEXT E
1700 M=0
1710 FOR NP=1 TO BP
1720 IF NP>1 THEN 1800
1730 FOR NC=1 TO AC+1
1740 M=M+1
1750 MFON(NC)=KS(M)+ES(1)*KD(1,N
C)
1760 PRINT "MOMENTI ALL'INCASTRO
 CON LA FONDAZIONE:COLONNA N.";N
C:MFON(NC)
1770 NEXT NC
1780 PRINT :::
1790 M=0
1800 PRINT "AL PIANO N.";NP
1810 FOR NC=1 TO AC+1
1820 M=M+1
1830 PRINT "NODO (";NP;"-";NC;")
"
1840 RE(M)=MD(M)+2*KE(M)+KO(M+1)

1850 IF M>(AC+1)THEN 1890
1860 RS(M)=2*KS(M)+ES(NP)*KD(NP,
NC)
1870 N(NP,NC)=-(RS(M)+MFON(NC))/
HP(NP)
1880 GOTO 1910
1890 RS(M)=2*KS(M)+KN(M-(AC+1))+
ES(NP)*KD(NP,NC)
1900 N(NP,NC)=-(RS(M)+RN(M-(AC+1
)))/HP(NP)
1910 RO(M)=MS(M)+2*KO(M)+KE(M-1)

1920 RN(M)=2*KN(M)+KS(M+AC+1)+ES
(NP+1)*KD(NP+1,NC)
1930 PRINT "(";NC;"-";NC-1;")";T
AB(13);"=";RO(M)
```

TI 99/4A

Seguito listato 1.

```
1940 PRINT "(";NC;"- SUD";")";TA
B(13);"=";RS(M)
1950 PRINT "(";NC;"-";NC+1;")";T
AB(13);"=";RE(M)
1960 PRINT "(";NC;"- NORD";")";T
AB(13);"=";RN(M)
1970 PRINT "TAGLIO SUL PILASTRO
N.";NC;TAB(13);"=";N(NP,NC)
1980 IF NC=1 THEN 2210
1990 TD(M-1)=TD(M-1)-(RE(M-1)+RO
(M))/LC(NC-1)
2000 TS(M)=TS(M)+(RE(M-1)+RO(M))
/LC(NC-1)
2010 PRINT "TAGLIO SULLA TRAVE";
NC-1;"d";TAB(13);"=";TD(M-1)
2020 IF Q=1 THEN 2200
2030 IF ASS(M-1)<>0 THEN 2070
2040 IE=TD(M-1)/QL(M-1)
2050 MPOS(M-1)=TD(M-1)*IE-QL(M-1
)*IE^2/2+MD(M-1)
2060 GOTO 2140
2070 FOR IE=.1 TO LC(NC-1)STEP 0
.01
2080 IF IE<ASS(M-1)THEN 2090 ELS
E 2110
2090 U=TD(M-1)-QL(M-1)*IE
2100 IF U<=0 THEN 2140 ELSE 2130

2110 U=TD(M-1)-QL(M-1)*IE-F(M-1)

2120 IF U<=0 THEN 2170
2130 NEXT IE
2140 PRINT
2150 MPOS(M-1)=TD(M-1)*IE-QL(M-1
)*IE^2/2+RE(M-1)
2160 GOTO 2190
2170 PRINT
2180 MPOS(M-1)=TD(M-1)*IE-QL(M-1
)*IE^2/2-F(M-1)*(IE-ASS(M-1))+RE
(M-1)
2190 PRINT "M POS.(";M-1;")";TAB
(13);"=";MPOS(M-1):"A M.";IE;"DA
 SINISTRA"
2200 PRINT :
2210 PRINT "TAGLIO SULLA TRAVE";
NC;"s";TAB(13);"=";TS(M)
2220 BREAK
2230 NEXT NC
2240 NEXT NP
2250 IF Q=1 THEN 2380
2260 PRINT :::
2270 M=BP*(AC+1)-(AC+1)
2280 FOR NP=BP TO 1 STEP -1
2290 PRINT "--------------------
--------"
2300 PRINT "AL PIANO";NP
2310 FOR NC=1 TO AC+1
2320 M=M+1
2330 EI(NC)=EI(NC)+TS(M)+TD(M)
2340 PRINT "COLONNA N.";NC;EI(NC
)
2350 NEXT NC
2360 M=M-2*(AC+1)
2370 NEXT NP
2380 STOP
2390 KE(M)=ACC*RE(M)
2400 KN(M)=ACC*RN(M)
2410 KS(M)=ACC*RS(M)
2420 KO(M)=ACC*RO(M)
2430 RETURN
```

Una partita a golf

Una simpatica simulazione, per il VIC inespanso, del famoso gioco di origine inglese. Campi, buche, angolazioni di tiro e velocità della pallina concentrati in questo programma, che saprà certamente impegnarvi.

di **Davide Sarli**

A chi non è mai venuto il desiderio di giocare una partita a golf? Il programma presentato, se pure non consente di sentire l'erba sotto i piedi, permette di soddisfare tale desiderio, attarverso una simulazione del gioco, completa di effetti grafici e sonori. È stata realizzata appositamente per il VIC 20 nella versione inespansa e non richiede l'uso di paddle o joystick. Il gioco rende invece necessarie buone doti di orientamento e di misura.

Le regole sono molto semplici. Il programma visualizza di volta in volta la posizione della pallina e quella della buca da raggiungere, con tanto di bandierina annessa. Per ogni colpo viene prima richiesta l'angolazione che si vuole dare al tiro, compresa tra 0 e 360 gradi rispetto alla linea dell'orizzonte, e poi la velocità con cui si farà viaggiare la pallina, che può variare da un minimo di uno ad un massimo di dieci. Dopo ogni colpo viene rappresentata la situazione aggiornata comprendente, oltre alla nuova posizione della pallina, anche il numero della buca che si vuole realizzare ed il numero di tiri già effettuati. Il percorso completo comprende nove buche e la presenza di funzioni casuali all'interno del programma lo rende sempre diverso ad ogni nuova partita. La realizzazione di ogni buca è accompagnata da un simpatico motivetto. A fine partita si ottiene il numero totale dei tiri impiegati per coprire l'intero percorso.

Due parole sulla strategia da usare per i principianti o per coloro che non hanno grossa dimestichezza con gli angoli! Conviene portarsi in prossimità di ogni buca attraverso spostamenti orizzontali o verticali, per poi tentare il tiro a buca quando, ormai vicini, è sufficiente utilizzare velocità pari a tre o meno.

Chi invece possiede spiccate doti di percezione spaziale può immediatamente puntare a buca. Si tenga però conto del fatto che, se la pallina è molto distante dalla buca, anche utilizzando la massima velocità ed imbroccando la giusta angolazione non si riesce a realizzare la buca con un colpo solo.

Il nostro record personale consiste in 15 tiri per le nove buche: vi invitiamo a superarlo. Si imbracci la mazza giusta, dunque, e buona fortuna.

```
0 GOSUB101
1 PRINT"[<1CLR>]"
2 POKE36879,221:ZT=1:ZS=1
3 DEFFNSI(X)=SIN((/2)/(90/X))
4 U1=4*(PEEK(36866)AND128)+37888
5 U2=4*(PEEK(36866)AND128)+64*(PEEK(36869
)AND120)
6 X2=INT(RND(1)*20)+1
7 Y2=INT(RND(1)*21)+1
8 R1=484-(Y2*22)+X2
9 IFR1<22THENR=22
10 IFR1>505THENR=505
11 POKEU1+R1,6:POKEU2+R1,81
12 X1=INT(RND(1)*18)+1
13 Y1=INT(RND(1)*18)+1
```

Listato 1 - Il programma BASIC.

Seguito listato 1.

```
14 IFX1=X2ANDY1=Y2THEN12
15 IFY1=Y4ANDX1=X4THEN12
16 R=484-(Y1*22)+X1
17 IFR<22THENR=22
18 IFR>505THENR=505
19 IFPEEK(U2+R)<>32ORPEEK(U2+R+1)<>32ORPE
EK(U2+R+22)<>32ORPEEK(U2+R+23)<>32THEN12
20 IFPEEK(U2+R-22)<>32ORPEEK(U2+R-44)<>32
ORPEEK(U2+R-43)<>32THEN12
21 POKEU2+R,85:POKEU2+R+1,73:POKEU2+R+22,
74:POKEU2+R+23,75
22 POKEU1+R,0:POKEU1+R+1,0:POKEU1+R+22,0:
POKEU1+R+23,0
23 POKEU2+R-22,103:POKEU2+R-44,103:POKEU2
+R-43,105
24 POKEU1+R-22,2:POKEU1+R-44,2:POKEU1+R-4
3,2
25 R1=484-(Y2*22)+X2
26 IFR1<22THENR=22
27 IFR1>505THENR=505
28 POKEU1+R1,6:POKEU2+R1,81
29 W=0:INPUT"[<1HOME>][<1BLK>]ANGOLAZIONE
";W$:W=VAL(W$)
30 IFW>360ORW<0THEN29
31 IFW=0THENW=0.000001
32 PRINT"[<1HOME>]                      "
33 I=0:INPUT"[<1HOME>][<1BLK>]VELOCITA'";
I$:I=VAL(I$)
34 IFI<1ORI>10THEN33
35 PRINT"[<1HOME>]                       "
36 POKE36878,15:POKE36877,200:FORZ=1TO150
:NEXT:POKE36877,0:POKE36878,0
37 Y3=(FNSI(W))*I
38 X3=SQR(I*I-Y3*Y3)
39 IFW>90ANDW<270THENX3=X3*(-1)
40 Y4=INT(Y3+Y2)
41 IFY4>22THENY4=22
42 IFY4<1THENY4=1
43 X4=INT(X3+X2)
44 IFX4>21THENX4=21
45 IFX4<1THENX4=1
46 POKEU2+R1,32
47 POKE36878,15
48 FORZ1=160TO170+I*2
49 POKE36876,Z1
50 FORZ2=1TOI*5:NEXT
51 NEXT
52 FORZ1=170+I*2TO160STEP-1
53 POKE36876,Z1
54 FORZ2=1TOI*5:NEXT
```

Seguito listato 1.

```
55 NEXT
56 POKE36876,0
57 FORZ=1TOI/3+1
58 POKE36874,200
59 FORM=1TO30-Z1*5:NEXT
60 POKE36874,0
61 FORM=1TO100:NEXT
62 NEXT
63 POKE36874,0:POKE36878,0
64 IFX4=X1ANDY4=Y1THEN69
65 IFX4=X1+1ANDY4=Y1THEN69
66 IFX4=X1ANDY4=Y1-1THEN69
67 IFX4=X1+1ANDY4=Y1-1THEN69
68 GOTO87
69 Y2=Y4:X2=X4:RESTORE
70 R1=484-(Y2*22)+X2
71 IFR1<22THENR=22
72 IFR1>505THENR=505
73 POKEU1+R1,6:POKEU2+R1,81
74 POKE36878,15
75 READA8,A9
76 IFA8=-1THEN81
77 POKE36876,A8:PRINT"[<1HOME>][<8CRSR D>
][<7CRSR R>][<1CHR$(213)>][<6CHR$(192)>][
<1CHR$(201)>]":PRINT"[<7CRSR R>][<1CHR$(1
94)>][<1RVS>]BUCA ![<1RVS OFF>][<1CHR$(
200)>]":PRINT"[<7CRSR R>][<1CHR$(202)>][<
6CHR$(192)>][<1CHR$(203)>]"
78 FORB9=1TOA9:NEXT
79 POKE36876,0
80 GOTO75
81 POKE36878,0:PRINT"[<1HOME>][<8CRSR D>]
[<7CRSR R>]          ":PRINT"[<7CRSR R>]
      ":PRINT"[<7CRSR R>]          "
82 ZT=ZT+1:ZS=ZS+1:Y2=Y4:X2=X4:POKEU2+31,
32
83 PRINT"[<1HOME>][<22CRSR D>]  BUCA";ZT;
TAB(13)"TIRO";ZS;"
84 GOSUB96
85 IFZT<=9THEN12
86 GOTO91
87 ZS=ZS+1:Y2=Y4:X2=X4
88 PRINT[<3CRSR U>]"[<1HOME>][<22CRSR D>]
  BUCA";ZT;TAB(13)"TIRO";ZS;"[<3CRSR U>]"
89 POKEU2+R1,32
90 GOTO25
91 PRINT"[<1CLR>]":ZS=ZS-1
92 PRINT"[<3CRSR D>][<1BLK>] BENE! HAI RE
ALIZZATO":PRINT"[<1CRSR D>]      9 BUCHE
IN"
```

Seguito listato 1.

```
93 PRINT"[<3CRSR D>]        "ZS;" TIRI"
94 PRINT"[<3CRSR D>] CERCA DI MIGLIORARE"
:PRINT"[<1CRSR D>]  QUESTO RISULTATO!"
95 PRINT"[<3CRSR D>]":END
96 POKEU2+R,32:POKEU2+R+1,32:POKEU2+R+22,
32:POKEU2+R+23,32
97 POKEU2+R-22,32:POKEU2+R-44,32:POKEU2+R
-43,32
98 RETURN
99 DATA201,500,191,500,201,500,191,500,19
5,260,191,200
100 DATA195,260,201,260,191,260,215,260,2
01,500,-1,-1
101 POKE36879,26
102 PRINT"[<1CLR>]   -[<1CHR$(209)>]-[<1C
HR$(215)>]-[<1CHR$(209)>][<1RVS>]GOLF[<1
RVS OFF>]-[<1CHR$(209)>]-[<1CHR$(215)>]-
[<1CHR$(209)>]-"
103 PRINT"[<2CRSR D>]DEVI REALIZZARE 9
104 PRINT"[<1CRSR D>]BUCHE CON IL MINORE
105 PRINT"[<1CRSR D>]NUMERO DI TIRI
106 PRINT"[<2CRSR D>]PER OGNI[<1CRSR R>]T
IRO DEVI
107 PRINT"[<1CRSR D>]SPECIFICARE L'ANGOLO
108 PRINT"[<1CRSR D>](0-360) E LA VELOCIT
A'
109 PRINT"(1-10)
110 PRINT"[<1CRSR D>]  [<1CHR$(213)>][<16
CHR$(192)>]111 PRINT[<1CHR$(201)>]"  [<1C
HR$(194)>] PREMI UN TASTO [<1CHR$(200)>]
   [<1CHR$(194)>]  PER GIOCARE   [<1CHR$(
200)>]"
112 PRINT"  [<1CHR$(202)>][<16CHR$(192)>]
113 GETA$:IFA$=[<1CHR$(203)>]""THEN113
114 RETURN
```

Analisi numerica

di **Pierluigi Adami**

Qualunque studente di ingegneria, o di matematica, che si sia imbattuto nello studio dell'analisi numerica non avrà potuto fare a meno di pensare che questa disciplina sia nata insieme con il primo calcolatore.
Anche se ciò non è vero, è però innegabile che proprio con il calcolatore i metodi dell'analisi numerica trovano piena ragion d'essere. Essi sono, infatti, iterativi: una serie di operazioni viene ripetuta finchè non si pervenga ad un risultato ottimo secondo un certo criterio. Ad esempio, si può verificare se il valore ottenuto all'iterazione i-esima differisca da quello ottenuto nell'iterazione precedente, a meno di un valore costante prefissato (criterio di arresto delle iterazioni). Questo sarà tanto più piccolo quanta maggiore è la precisione che si desidera ottenere. In questo programma il criterio di arresto è 1E-6, che garantisce una buona precisione in un numero limitato di iterazioni.
Se l'errore non riesce a scendere al di sotto del criterio di arresto, si dice che il metodo non converge.
Ciò detto, veniamo al programma vero e proprio: esso è in realtà composto da quattro programmi distinti (separabili facilmente) e da due sottoprogrammi.

Programma soluzione equazioni

Esso sfrutta i due metodi classici delle secanti per trovare una radice interna ad un intervallo (figura 1).
È ovviamente indispensabile definire l'intervallo in cui si ritiene giaccia la radice che interessa; la convergenza (ossia l'aver trovato una stima della radice reale, che differisca da questa meno dell'errore previsto, in un numero limitato di iterazioni) dipende dall'andamento della funzione nell'intervallo scelto. Il criterio di arresto è in questo caso 1E-8.
Per garantire la convergenza si può consigliare:
- cercare, se possibile, un intervallo in cui la funzione volga sempre la concavità dalla stessa parte (ossia, sempre verso l'alto o il basso);
- cercare, se possibile, di definire l'intervallo in modo che la funzione assuma segno opposto nei due estremi;
- evitare, ovviamente, punti di discontinuità interni all'intervallo;
- evitare radici multiple.

Anche se uno di questi punti non fosse rispettato, si può ugualmente avere la convergenza. Talvolta il metodo trova la radice più vicina all'intervallo definito, anche se è al di fuori di esso.
La formula iterativa è:

$$x(I+1) = x(I) - F(x(I)) \frac{x(I) - x(I-1)}{F(x(I)) - F(x(I-1))}$$

Figura 1 - Metodo delle secanti per la ricerca di una radice di un'equazione.

Figura 2 - Approssimazione di integrali tramite somma di trapezi.

Il primo valore x(0) da cui parte il procedimento è l'estremo opposto di intervallo verso cui la curva mostra la sua convessità (estremo di Fourier), coincidente con (a) nel 1° metodo delle secanti, e con (b) nel 2° metodo.
(Il 2° metodo sfrutta una formula leggermente modificata).
Il programma prova a risolvere il problema provando entrambi i metodi in sequenza. Se, nonostante ciò, non riesce a convergere, stampa un risultato parziale, da valutare con cautela: esso può anche essere molto prossimo alla radice reale e può dunque servire ad orientare meglio l'intervallo nel successivo tentativo, ma può anche discostarsi di molto.

Programma soluzione sistemi lineari

Il metodo usato per risolvere il problema è quello classico delle eliminazioni successive di Gauss. Si rimanda ad un testo specializzato per una trattazione dettagliata.
Esso comunque sfrutta un principio ben noto: se in una matrice (o array, o schiera) si sostituiscono a righe (o colonne) delle loro combinazioni lineari (ad esempio si somma ad una riga il valore di un'altra moltiplicato per due) il determinante della matrice non cambia.
Molti ricorderanno il metodo di Cramer per la soluzione dei sistemi lineari: ebbene, in quel metodo era proprio il determinante della matrice dei coefficienti a svolgere il ruolo maggiore nel calcolo delle radici.
Allora, il metodo di Gauss, lasciando inalterato il determinante della matrice dei coefficienti, opera successive elaborazioni per semplificare il sistema. Passo dopo passo, giungerà proprio alle soluzioni con estrema rapidità.
Dato che questo metodo richiede dei rapporti con dei coefficienti via via calcolati, c'è il rischio che l'errore di calcolo si moltiplichi, quando i coefficienti diventino troppo piccoli.
Questi parametri sono il pivot (vedi istruzione 580) ed il coefficiente A (n, n) della matrice (n è il numero equazioni del sistema), che vengono modificati ad ogni iterazione.
Se questi valori scendono al di sotto di 1E-6, allora il programma si arresta.
Questo è il caso dei sistemi detti mal-condizionati e di quelli con det (A) = 0.
Il programma provvede comunque a visualizzare i valori del pivot e l'ultimo valore di A (n, n), insieme con gli indici di iterazione, prima dei risultati. Se si osservano valori molto piccoli di uno dei parametri (meno di 1E-3), bisogna comunque valutare con cautela i risultati per la presenza di (eventuali) moltiplicazioni d'errore.

Programma soluzione integrali definiti

I metodi di calcolo numerico degli integrali, sfruttano un principio di approssimazione piuttosto semplice: sostituiscono alla curva da integrare nell'intervallo (a, b) una spezzata (vedi figura 2): l'integrale definito della funzione, che è numericamente pari all'area descritta dalla curva con l'asse x, può quindi essere approssimato dalla somma delle aree dei vari trapezi, costruiti come in figura 2.
In pratica si sostituisce ad ogni intervallo un polinomio interpolatore: nel caso dei trapezi, col polinomio più semplice, la retta. È naturale che aumentando il numero dei trapezi, ossia stringendo gli intervalli Dx, si migliora l'approssimazione.
Il metodo che viene qui presentato è il metodo delle parabole, del secondo ordine.
Senza soffermarsi ulteriormente, sono comunque da evidenziare alcune considerazioni sull'errore:

Figura 3 - Curva che interpola un insieme di punti.

Listato 1 - Il listato del programma in questione.

```
1 REM **********************************
2 REM *                                *
3 REM * ANALISI NUMERICA:              *
4 REM *                                *
5 REM * 1) SOLUZ.EQUAZIONI             *
6 REM * 2) SOL.SISTEMI LINEARI         *
7 REM * 3) SOL.INTEGRALI DEFINITI      *
8 REM * 4) INTERPOLAZIONE MIN.QUADR. *
9 REM **********************************
10 REM *                               *
11 REM * PER COMMODORE 64              *
12 REM *                               *
13 REM * DI:PIERLUIGI ADAMI            *
14 REM * V.GIUNIO SILANO 18,00174 ROMA*
15 REM *                               *
16 REM *********************************
17 POKE53280,6:POKE53281,6:PRINT"[<1YEL>]
"
18 PRINT"[<1CLR>]CERCO L'IND.MEMORIA DELL
A ISTRUZ.N.3012"
19 FORI=7800TO10000:IFPEEK(I)+256*PEEK(I+
1)=3012THEN21
20 NEXT:END
21 CM=I+9:C$=STR$(CM):TY=61
22 FORJ=1TO4:POKEI+J+78,ASC(MID$(C$,J+1,1
)):NEXT
23 PRINT"[<1CRSR D>]L'ISTRUZ.3012 INIZIA
ALL'INDIR.NUM:";I:GOTO4000
24 REM
25 PRINT"[<1CLR>]MENU'DISPONIBILE IN QUES
TO PROGRAMMA":PRINT:PRINT:PRINT
28 PRINT"SOLUZIONE EQUAZIONI...........1"
30 PRINT:PRINT"SOL.SISTEMI LINEARI.......
....2"
40 PRINT:PRINT"CALCOLO INTEGRALI.........
....3"
50 PRINT:PRINT"INTERPOLAZIONE...........
....4"
60 PRINT"[<3CRSR D>]PREMI IL NUM.DESIDERA
TO"
96 GETN$:IFN$=""THEN96
97 IFVAL(N$)<1ORVAL(N$)>4THEN96
98 VL=VAL(N$):ONVLGOTO100,500,800,1000,20
00
99 REM
100 PRINT"[<1CLR>]PROGRAMMA SOLUZIONE EQU
AZIONI"
103 GOSUB3000
106 PRINT"[<1CLR>]DEFINISCI L'INTERVALLO
OVE SI[<1CRSR D>]"
107 PRINT"RITIENE RISIEDA LA RADICE[<3CRS
```

Seguito listato 1.

```
R D>]"
110 INPUT"ESTREMO INFERIORE";A:PRINT
120 INPUT"ESTREMO SUPERIORE";B
140 X=B:L=0
144 REM
145 REM ORA INIZIA LE ITERAZIONI CON
146 REM IL I METODO DELLE SECANTI
147 REM
150 FORI=1TO30
160 N=A*FNA(X)-X*FNA(A)
162 D=FNA(X)-FNA(A)
163 IFD=0ANDL=0THENPRINT"[<2CRSR D>]INDET
ERMINAZIONE":GOTO300
164 IFD=0ANDLTHEN200
166 R=N/D
170 E=ABS(R-X)
180 IF E<=1E-8THEN220:REM CRITERIO ARREST
O
190 X=R:NEXT:IFL=0THEN300
200 PRINT"[<2CRSR D>]NON CONVERGE[<1CRSR
D>]"
205 PRINT"DIFF.ULTIMI 2 VALORI";E
210 PRINT"[<1CRSR D>]RISULTATO PARZ.=";R
212 PRINT"[<1CRSR D>]CAMBIO INTERVALLO? (
S/N)"
214 GETZ$:IFZ$=""THEN214
216 IFZ$="S"THENPRINT"[<1CLR>]":GOTO110
218 GOTO420
220 PRINT"[<2CRSR D>]N.ITERAZIONI";I:PRIN
T"[<3CRSR D>]LA RADICE DELL'EQUAZ.E'";R:G
OTO420
300 PRINT"[<4CRSR D>]PROVO II METODO[<3CR
SR D>]":REM SECONDO METODO DELLE SECANTI
400 X=A:A=B:N=-N:D=-D:L=1
410 GOTO150
420 REM AZZERA LA FUNZIONE E DA'IL MENU
425 GOTO4000
```

- l'ampiezza degli intervalli Dx di divisione dell'intervallo di integrazione (a, b) aumenta all'aumentare dello stesso (a, b): dunque, aumentando l'intervallo d'integrazione aumenta di approssimazione. Perciò un intervallo troppo ampio può portare ad una non-convergenza del metodo, o ad una convergenza molto lenta;
- evidenziamo allora la formula analitica dell'errore:

$$\text{ERRORE} = \frac{(b-a)\ 15}{2880 \star N} \star F\,(IV)\,(@)$$

ove F (IV) (@) è la derivata quarta della funzione in un punto interno ad (a, b), N è il numero di intervallini Dx elevato alla quarta.
Tutto ciò, in pratica, significa che se si vuole garantire la convergenza, bisogna non allargare troppo (a, b); è però probabile la convergenza nel caso in cui il valore N e la derivata quarta siano tali da mantenere l'errore comunque piccolo.

In caso di mancata convergenza (errore > 1E-6), il programma rende noti i valori parziali cui è giunto nell'ultima iterazione.
Il risultato parziale ottenuto potrà essere allora valutato in base ai valori dell'errore visualizzati. Viene anche fornito l'errore percentuale (stimato): è chiaro che, se esso supera il 100% vuol dire che il risultato ottenuto è ben lontano dalla convergenza.
Il tempo di calcolo può essere notevole, se la funzione da integrare è complessa.

Programma interpolazione minimi quadrati

Il problema classico dell'interpolazione è tipico delle scienze statistiche: ottenuto un certo insieme di punti come risultato dell'osservazione di

Seguito listato 1.

```
490 REM
498 REM SISTEMI EQUAZIONI
499 REM
500 PRINT"[<1CLR>]PROGRAMMA SOL.SISTEMI L
INEARI[<4CRSR D>]"
505 INPUT"NUM.EQUAZIONI";N:PRINT
506 IFN<2ORN>80THEN505
507 DIMA(N,N+1),X(N),I(N,N)
520 NP=N+1:NM=N-1
530 E=1E-6
540 PRINT"INSERISCI MATRICE COMPLETA PER
RIGHE":PRINT
545 PRINT"M.COMPLETA=(MAT.COEFF.:VETT.TER
M.NOTI)[<1CRSR D>]"
550 FORI=1TON:FORJ=1TON+1
555 IFJ=N+1THENPRINT"=";:GOTO560
557 PRINT"A(";I;",";J;")=";
560 INPUTA(I,J):PRINT:NEXT:NEXT
570 FORK=1TONM
580 PV=ABS(A(K,K)):REM CALCOLA IL PIVOT D
ELLE ELIMINAZIONI DI GAUSS
590 KM=K:KP=K+1
595 FORI1=KPTON
600 IFPV>=ABS(A(I1,K))THEN610
605 PV=ABS(A(I1,K)):KM=I1:REM AGGIORNA IL
 PIVOT
610 NEXT
620 IFVL=2THENPRINT:PRINT"INDICE:";K,"PIV
OT:";PV
625 IFPV<=ETHEN725:REM PIVOT TROPPO PICCO
LO
630 IFKM=KTHEN650
635 FORJ1=KTONP:T=A(K,J1)
640 A(K,J1)=A(KM,J1)
645 A(KM,J1)=T:NEXT
650 FORI=KPTON:FORJ=KPTONP
655 A(I,J)=A(I,J)-(A(I,K)/A(K,K))*A(K,J)
```

una popolazione (campione), può essere utile avere a disposizione una curva che descriva nel modo migliore possibile l'andamento di quei punti. Avere a disposizione una funzione che approssimi i punti dell'esperimento diventa utilissimo quando si voglia manipolare i dati, cercandone particolari caratteristiche, o nel problema della predizione. Ricavato, ad esempio, come dato il valore medio dei prezzi di una certa merce nei mesi di un anno, una volta ottenuta la curva interpolante si potrà predire l'andamento futuro del mercato.
È ovvio, però, che l'interpolazione riveste una notevole importanza in altre discipline, oltre alla statistica.
Il problema sta nel cercare la curva ottima. Si deve dunque definire un parametro che valuti la bontà del risultato raggiunto. Il parametro più spesso usato è la somma degli errori tra i valori effettivi e quelli stimati con la funzione, elevati al quadrato (errore quadratico).
Tipi diversi di funzioni possono dunque approssimare più o meno bene, secondo l'errore quadratico, l'andamento dei punti dell'esperimento.

Il programma permette all'operatore di scegliere tra tre tipi di curve: esponenziale, potenza, polinomiale. La scelta può essere fatta in base ad una preventiva osservazione dell'andamento dei punti, o in base alle particolari esigenze che si vogliono dalla stima.
Dopo aver fornito i dati in ingresso, ed una rapidissima elaborazione, viene visualizzata l'equaz. della funzione, i valori inseriti in input ed i valori stimati ottenuti con la funzione interpolatrice. Subito dopo compare l'errore quadratico, che ci può servire per valutare la bontà della stima otte-

Seguito listato 1.

```
660 NEXT:NEXT:NEXT
661 REM
662 REM ECCO RISULTATI E DATI UTILI
663 REM
665 IFVL=2THENPRINT:PRINT"N.RIGHE:";N,"A(
N,N):";A(N,N):PRINT:PRINT
670 IFABS(A(N,N))<=ETHEN725
675 X(N)=A(N,NP)/A(N,N)
680 FORM=1TONM
690 I2=N-M:SM=0
700 FORJ2=I2+1TON
705 SM=SM+A(I2,J2)*X(J2):NEXT
710 X(I2)=(A(I2,NP)-SM)/A(I2,I2):NEXT
711 IFVL=4THENRETURN
720 PRINTTAB(10);"ECCO I RISULTATI":PRINT
725 FORI2=1TON:PRINTTAB(10);"X(";I2;")=";
X(I2):NEXT:GOTO4000
799 REM
800 PRINT"[<1CLR>]PROGRAMMA CALCOLO INTEG
RALE"
801 REM
802 REM CALCOLO INTEGRALI DEFINITI
803 REM CON IL METODO DELLE PARABOLE
817 GOSUB3000
818 PRINT"[<1CLR>]":INPUT"ESTR.INFER.INTE
GR.";A
819 PRINT:INPUT"ESTR.SUPER.INTEGR.";B:PRI
NT
850 MM=25:E=1E-6:REM NUM MAX ITERAZIONI E
 CRITERIO D'ARRESTO DELL'ERRORE
860 VP=(FNA(A)+4*FNA((A+B)/2)+FNA(B))*(B-
A)/6
870 VZ=FNA(A)+FNA(B)
880 FORM=2TOMM
890 H=(B-A)/(2*M):M2=2*M-1:IC=4:VS=0
900 FORI=1TOM2
910 VS=VS+IC*FNA(A+I*H)
920 IC=6-IC:NEXT
930 VS=(VZ+VS)*H/3
940 IFABS(VS-VP)<=ETHEN985
950 ER=ABS(VS-VP):VP=VS:NEXT
960 PRINT"NUM.ITERAZ.> DEL MASSIMO[<2CRSR
 D>]"
964 PRINT"DIFF.ULTIMI 2 VALORI:";ER
965 PRINT"[<1CRSR D>]ERRORE PROBAB.CIRCA:
";1130*ER"%"
970 PRINT"[<2CRSR D>]RISULTATO PARZIALE="
;VS
975 GOTO992
985 PRINT"NUM.ITERAZIONI:";M
987 PRINT:PRINT"RISULT.INTEGRALE=";VS
```

Seguito listato 1.

```
992 GOTO4000
1000 REM
1001 PRINT"[<1CLR>]PROGRAMMA INTERP.MIN.Q
UADRATI"
1002 REM
1010 DIMX1(100),Y1(100)
1015 PRINT"[<4CRSR D>]PONI N=0 PER UNA F.
POTENZA"
1020 PRINT"[<1CRSR D>][<5CRSR R>]N=1 PER
UNA F.ESPONENZIALE"
1025 PRINT"[<3CRSR D>]PER UNA F.POLINOMIA
LE,N=GRADO POLIN.+1"
1030 INPUT"[<2CRSR D>]VALORE N";N1
1031 N2=N1:IFN2>=2THEN1033
1032 N2=2
1033 DIMA1(N2,N2),D(N2),X(N2),A(N2,N2+1)
1035 INPUT"[<1CLR>]NUM.PUNTI DA INTERPOLA
RE";M1
1040 PRINT"[<2CRSR D>]INSERISCI I VALORI
ORDINATI[<2CRSR D>]"
1044 REM INPUT DEI DATI
1045 FORI=1TOM1
1050 PRINT"X(";I;")=";:INPUTX1:X1(I)=X1
1055 PRINTTAB(14);"[<1CRSR U>]Y(";I;")=";
:INPUTY2:Y1(I)=Y2:NEXT
1125 REM ** CALCOLA LOG.DEI DATI PER LE
1126 REM ** F.POTENZA ED ESPONENZIALE
1130 IFN1>=2THEN1170
1132 FORI=1TOM1:IFY1(I)<=0THEN1700
1135 Y1(I)=LOG(Y1(I)):NEXTI
1150 IFN1=1THEN1170
1152 FORI=1TOM1:IFX1(I)<=0THEN1700
1155 X1(I)=LOG(X1(I)):NEXTI
1170 REM
1210 FORI=1TON2:FORJ=1TON2
1220 IF(I+J)>2THEN1235
1225 A1(I,J)=M1
1230 GOTO1250
1235 FORK=1TOM1:A1(I,J)=A1(I,J)+X1(K)^(I+
J-2)
1245 NEXTK
1250 NEXTJ
1255 FORK=1TOM1
1260 IFI>1THEN1275
1265 D(I)=D(I)+Y1(K)
1270 GOTO1280
1275 D(I)=D(I)+Y1(K)*X1(K)^(I-1)
1280 NEXTK:NEXTI
1310 N=N2:NM=N-1:NP=N+1:E=1E-6
1320 FORI=1TON:FORJ=1TON:A(I,J)=A1(I,J):N
EXTJ:NEXTI
```

Seguito listato 1.

```
1330 FORI=1TON:A(I,N+1)=D(I):NEXT
1331 REM
1332 REM RISOLVE IL SISTEMA LINEARE
1333 REM USANDO IL PROGRAMMA APPOSITO
1334 REM
1340 VL=4:GOSUB570:REMCHIAMATA AL PROGRAM
MA SOLUZIONE SISTEMI LINEARI
1395 IFN1>1THEN1430
1400 C1=EXP(X(1))
1405 IFN1=1THEN1420
1406 REM STAMPA LA FUNZ.INTERPOLANTE
1410 PRINT"[<1CLR>]FUNZ.POTENZA:":PRINT"[
<1CRSR D>]Y=";C1;"*X^";X(2)
1415 GOTO1490
1420 PRINT"[<1CLR>]FUNZ.ESPONENZ.:":PRINT
"[<1CRSR D>]Y=";C1;"*EXP(";X(2);"*X)"
1425 GOTO1490
1430 IFX(2)>=0THEN1445
1435 PRINT"[<1CLR>]F.POLINOMIALE:":PRINT"
[<1CRSR D>]Y=";X(1);X(2);"X";
1440 GOTO1450
1445 PRINT"[<1CLR>]F.POLINOMIALE:":PRINT"
[<1CRSR D>]Y=";X(1);"+";X(2);"*X";
1450 IFN1=2THEN1485
1455 FORI=3TON1
1460 IFX(I)>=0THEN1475
1465 PRINTX(I);"*X^";I-1;
1470 GOTO1480
1475 PRINT"+";X(I);"*X^";I-1;
1480 NEXTI
1485 PRINT
1490 REM
1495 REM ORA CALCOLA L'ERRORE
1500 REM
1505 IFN1>=2THEN1545
1510 FORI=1TOM1
1515 Y1(I)=EXP(Y1(I))
```

nuta e confrontarla con altre.
Il programma è tratto dal libro: Programmare in BASIC edizioni Schaum, ma è stato notevolmente modificato per il Commodore 64: tra l'altro, dovendo risolvere un sistema lineare, nell'istruzione 1340 c'è una chiamata al programma Soluzione Sistemi Lineari, considerato in quel caso come un sottoprogramma.

Sottoprogramma definisci la funzione

Questo sottoprogramma permette all'utente di definire una funzione in maniera diretta, come se fosse un dato in INPUT.
L'istruzione BASIC DEFFNA (X) =, infatti, impone di inserire preventivamente la funzione nel programma, prima del RUN.
Ciò ovviamente costringe a modificare una linea del programma, ogniqualvolta si voglia cambiare la funzione. Procedura inaccettabile per un programma come questo, che prevede la possibilità di passare subito da un sottoprogramma all'altro tramite un menu.
Le istruzioni 3000 e seguenti consentono invece di memorizzare immediatamente la funzione: dopo aver premuto RETURN la funzione visualizzata viene subito utilizzata dal programma chiamante, e poi cancellata una volta eseguito il compito (le linee 4000 e seguenti "ripuliscono" la memoria).
Nell'istruzione:
3012 DEFFNA (X) =000000000000000000000000
00000000000000000000000000000000000
i 62 zeri servono a conservare 62 locazioni di memoria, pronte ad accettare la funzione effettiva

Seguito listato 1.

```
1520 NEXTI
1525 IFN1=1THEN1545
1530 FORI=1TOM1
1535 X1(I)=EXP(X1(I))
1540 NEXTI
1545 PRINT
1550 PRINT"[<3CRSR D>]X (REALE)";TAB(16);
"Y (REALE)";TAB(28);"Y (STIM.)[<1CRSR D>]
"
1555 S=0
1560 FORI=1TOM1
1565 IFN1>=2THEN1595
1570 IFN1=1THEN1585
1575 Y1=C1*X1(I)^X(2)
1580 GOTO1615
1585 Y1=C1*EXP(X(2)*X1(I))
1590 GOTO1615
1595 Y1=X(1)
1600 FORJ=2TON1
1605 Y1=Y1+X(J)*X1(I)^(J-1)
1610 NEXTJ
1615 S=S+(Y1(I)-Y1)^2
1620 PRINTX1(I);TAB(13);Y1(I);TAB(26);Y1
1625 NEXTI
1630 PRINT
1635 PRINT"SOMMA ERRORI QUADR.=";S
1645 GOTO4100
1700 PRINT"[<3CRSR D>]VAL.NEGATIVI O NULL
I IN INGRESSO[<1CRSR D>]"
1710 PRINT"USA L'APPR.POLINOMIALE[<3CRSR
D>]"
1720 GOTO4000
3000 REM
3005 REM SUBROUTINE DEF.FUNZIONE
3010 REM
3011 REM 62 ZERI DOPO FNA(X)=
3012 DEFFNA(X)=000000000000000000000000000
```

inserita con la GET (istruzione 3045). I valori inseriti in memoria non corrispondono sempre ai rispettivi valori ASCII: le funzioni di libreria ed i simboli algebrici hanno un loro codice particolare. Ad esempio la funzione SIN (x) è identificata dal numero 191, corrispondente al valore ASCII del simbolo grafico ▚.
Nell'istruzione 3500 la POKE inserisce nelle locazioni di memoria uguali o successive a CM (indirizzo di memoria del primo zero dell'istruzione 3012) i valori ricavati dalla serie di IF (da 3050 in poi).
C'è però un problema: l'indirizzo di memoria del primo zero dell'istruzione 3012 può variare notevolmente, anche a seguito di piccole variazioni del programma. Chi copierà il programma probabilmente giungerà ad un valore di CM diverso da quello che compare nell'istruzione 3013 CM= ..., valore del tutto indicativo.

Comunque, è lo stesso Commodore che va a cercare "dove ha messo" l'istruzione 3012. Come molti sapranno, il numero di un istruzione è memorizzato in due celle consecutive e si ottiene così:
numero istruzione = PEEK (I) + 256 ★ PEEK (I + 1)
Le istruzioni 18-21 provvedono a cercare all'inizio del programma, una volta per tutte, il valore CM. Per risparmiare tempo la ricerca inizia alla locazione 7500: se si intende modificare notevolmente il programma, si può allargare il campo, setacciando al limite tutta la RAM del BASIC a partire dalla locazione numero 2048. Prima o poi l'istruzione 3012 verrà trovata.
Nel programma le istruzioni 3012 e 3013 non debbono essere modificate!
L'indirizzo trovato viene memorizzato in CM, ma

Seguito listato 1.

```
000000000000000000000000000000000000
3013 CM=8083:REM INDIRIZZO PRIMO ZERO
3015 PRINT"[<1CRSR D>]DEFINISCI LA FUNZIO
NE F(X)[<1CRSR D>]":TY=0
3020 PRINT"[<7CRSR R>]AVVERTENZA![<1CRSR
D>]"
3025 PRINT"UTILIZZA QUESTI NOMI:[<1CRSR D
>]"
3030 PRINT"A(X)=ATN(X)     M(X)=ABS(X)"
3031 PRINT"C(X)=COS(X)     Q(X)=SQR(X)"
3032 PRINT"D(X)=SGN(X)     R(X)=RND(X)"
3033 PRINT"E(X)=EXP(X)     S(X)=SIN(X)"
3034 PRINT"L(X)=LOG(X)     T(X)=TAN(X)[<1C
RSR D>]"
3035 PRINT"NELLA NOTAZ.SCIENTIF.DEI NUMER
I CON[<1CRSR D>]"
3036 PRINT"ESPONENTE, USA LETTERA P AL PO
STO DI E[<3CRSR D>]"
3040 PRINT"*** F(X)=[<1RVS>]>[<1CRSR L>]
[<1RVS OFF>]";
3045 GETZ$:IFZ$=""THEN3045
3046 IFASC(Z$)=13 THEN3600:REM TASTO RETU
RN
3047 IFASC(Z$)=222THENV=255:GOTO3500:REM
PI GRECO
3048 REM CORREGGE CON INST DEL
3049 IFPEEK(203)=0ANDTY>0THENTY=TY-1:POKE
CM+TY,48:PRINT"[<1CRSR L>]  [<2CRSR L>][<
1RVS>]>[<1CRSR L>][<1RVS OFF>]";:GOTO30
45
3050 IFZ$="X"THENV=88:GOTO3500
3051 IFZ$="+"THENV=170:GOTO3500
3055 IFZ$="-"THENV=171:GOTO3500
3060 IFZ$="*"THENV=172:GOTO3500
3064 IFZ$="^"THENV=174:GOTO3500
3065 IFZ$="/"THENV=173:GOTO3500
3066 IFZ$="S"THENV=191:GOTO3500
3070 IFZ$="C"THENV=190:GOTO3500
3075 IFZ$="L"THENV=188:GOTO3500
3080 IFZ$="E"THENV=189:GOTO3500
3085 IFZ$="A"THENV=193:GOTO3500
3090 IFZ$="T"THENV=192:GOTO3500
3095 IFZ$="M"THENV=182:GOTO3500
3100 IFZ$="Q"THENV=186:GOTO3500
3102 IFZ$="P"THENV=69:GOTO3500
3105 IFZ$="R"THENV=187:GOTO3500
3110 IFZ$="D"THENV=180:GOTO3500
3115 IFZ$="("THENV=40:GOTO3500
3120 IFZ$=")"THENV=41:GOTO3500
3125 IFASC(Z$)>47ANDASC(Z$)<58THENV=ASC(Z
$):GOTO3500
```

Seguito listato 1.

```
3130 GOTO3045
3500 PRINTZ$;"[<1RVS>]>[<1CRSR L>][<1RV
S OFF>]";:POKECM+TY,V:TY=TY+1:IFTY<=60THE
N3045
3505 PRINT"[<1CLR>]FUNZIONE TROPPO LUNGA"
:GOTO4000
3600 POKECM+TY,170:RETURN
4000 REM AZZERA ISTR.3012 E CLEAR
4010 FORMM=CMTOCM+TY:POKEMM,48:NEXT:TY=0
4100 PRINT"[<3CRSR D>]PREMI UN TASTO PER
IL MENU'"
4110 GETZ$:IFZ$=""THEN4110
4120 CLR:GOTO25
```

ancora non basta: dopo l'esecuzione dei programmi Sistemi Lineari o Interpolazione si è costretti ad usare l'istruzione CLR (vedi linea 4120) per evitare l'errore di ridimensionamento di Array in successive esecuzioni. Ciò però azzera tutte le variabili, alterando quindi anche il valore di CM = I + 9 (istruzione 103).
Anche in questo caso è il programma stesso che provvede a "bloccare" il valore di CM corretto. Infatti, le istruzioni 21 e 22 scrivono nelle appropriate locazioni di memoria il valore ricavato di CM, "ricostruendo" l'istruzione di assegnazione 3013 CM = ...

Dunque, qualunque sia il valore CM che verrà arbitrariamente inserito nella linea 3013 (di 4 cifre, però!) sarà lo stesso Commodore che provvederà, dopo il RUN, a correggere automaticamente il listato del programma con il valore CM corretto.
Se si vuole, si possono separare dal resto i programmi.

- Soluzione Equazioni

1) Eliminare linee: 103, 420, 425;
2) Porre 420 END;
3) Porre 103 DEFFNA (X) = ...

- Sistemi Equazioni

1) Eliminare linea 711;
2) Eliminare GOTO4000 dalla linea 725;
3) Eliminare IFVL = 2THEN dalla linea 620;
4) Eliminare IFVL = 2THEN dalla linea 665.

- Calcolo integrali

1) Eliminazione linee 804, 815, 817, 992;
2) Porre 815 DEFFNA (X) = ...

Questo programma può girare anche su VIC 20 espanso: il problema è sapere dove inizia la RAM BASIC. Una volta noto il valore, la ricerca della locazione CM inizierà con il FOR nella linea 18, a partire da quella locazione di memoria.
Va ovviamente anche modificata per il VIC 20 l'istruzione 17, che gestisce i colori del bordo e dello sfondo.